НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА СИСТЕМНОГО ПРОГРАМУВАННЯ І
СПЕЦІАЛІЗОВАНИХ КОМП'ЮТЕРНИХ СИСТЕМ

«На правах рукопису»
УДК _____

«До захисту допущено»

Завідувач кафедри СПСКС

_____ В.П.Тарасенко
(підпис) (ініціали, прізвище)

"___"_____2018р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 123 Комп'ютерна інженерія
(«Системне програмування»)

на тему: «МЕТОДИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЗАХИСТУ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ ВІДБИТКІВ ПАЛЬЦІВ У BLOCKCHAIN ПРОЕКТАХ»

Виконав: студент II курсу, групи КВ-62м
(шифр групи)

Пацьора Андрій Андрійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник д.т.н., професор Зайцев В.Г. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Системне програмування

ЗАТВЕРДЖУЮ
Завідувач кафедри СПСКС
_____ _В.П.Тарасенко__
(підпис) (ініціали, прізвище)

«___»_____2018р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Пацьори Андрія Андрійовича
(прізвище, ім'я, по батькові)

1. Тема дисертації «Методи підвищення ефективності захисту інформації з використанням відбитків пальців у blockchain проектах»,
науковий керівник дисертації д.т.н., професор Зайцев В.Г.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «22» березня 2018 р. №986-с

2. Термін подання студентом дисертації 11 травня 2018 р.

3. Об'єктом дослідження є методи автентифікації відбитків пальців для проектів blockchain, що використовують передачу великих масивів даних.

4. Предметом дослідження є методи виконання автентифікації відбитків та зіставлення по базі даних.

5. Перелік завдань, які потрібно розробити
- провести порівняльний аналіз існуючих методів.
- провести попередню обробку відбитків пальців.

- провести порівняльний аналіз нових вхідних даних відбитків .
- запропонувати алгоритм Bozorth як основний.
- інтегрувати методи у blockchain.
- відстежити та перевірити вхідні потоки даних за допомогою ідентифікації відбитків.
- провести експерименти на 40+ добровольцях та обробити дані з використанням запропонованих методів.
- провести аналіз отриманих результатів.

6. Перелік ілюстративного матеріалу

- Схема KD-дерева використаного у розробці
- Схема методу виявлення мініатюр на відбитках
- Схема операції у blockchain
- Схема реєстрації відбитків пальців
- Структурна схема градусного рівня синтезу
- Схема проведення транзакцій у blockchain проектах

7. Перелік публікацій

- Тези доповіді «Методи підвищення ефективності захисту інформації у blockchain проектах»
- Тези доповіді «Автентифікація відбитку пальців та інтеграція у blockchain»

8. Дата видачі завдання 5 вересня 2016 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Ґрунтовне ознайомлення з предметною областю дослідження	17.12.2016	
2	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури	04.03.2017	
3	Робота над першим розділом магістерської дисертації; проведення	07.05.2017	

	наукового дослідження		
4	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	21.09.2017	
5	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	02.12.2017	
6	Проведення наукового дослідження; робота над третім розділом магістерської дисертації; підготовка матеріалів доповіді на конференції ПМК-2018.	21.02.2018	
7	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу;	20.04.2018	
8	Оформлення текстової і графічної частини магістерської дисертації	21.04.2018	
9	Попередній розгляд магістерської дисертації на кафедрі	27.04.2018	

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Актуальність теми. Відбиток пальців – один із найпоширеніших біометричних способів для розпізнавання даних через надійність, швидкість та продуктивність. Відбитки залишаються незмінними протягом усього життя особи. Відповідно до цих переваг застосування біометричних відбитків пальців все більше використовується різними комерційними та державними організаціями. Функція ідентифікації відбитків пальців дозволяє автоматично та надійно витягати мінуси з вхідних зображень відбитків пальців. Проте ефективність алгоритму вилучення мінусів залежить від якості вхідних відбитків пальців. Щоб забезпечити надійність роботи системи перевірки автентичності за відбитками пальців, необхідно попередньо обробляти зображення. Ця теза описує кроки, що відбуваються під час попередньої обробки відбитків пальців, що покращує чіткість структури хребта та біфуркації вхідних відбитків пальців. Після попередньої обробки мінуси виводяться та зберігаються в базі даних. Далі впроваджено онлайн-систему автентифікації відбитків пальців, в якій використовується елементарна стратегія індексування. Індексування даних відбитків пальців здійснюється для ідентифікації та отримання невеликої підмножини даних кандидатів з бази даних. Експериментальні роботи показують, що інтеграція онлайн-системи алгоритму попередньої обробки покращує загальний час відгуку.

Об’єктом дослідження метод підвищення ефективності захисту інформації.

Предметом дослідження є методи підвищення ефективності захисту інформації з використанням ідентифікації відбитків пальців для проектів на blockchain.

Мета і задачі дослідження: Створити спосіб ефективного захисту інформації для проектів, котрі використовують блочну структуру з

використанням ідентифікації відбитків пальців на будь яких етапах запису інформації.

Провести експерименти за алгоритмами обраного способу по ідентифікації різних типів відбитків пальців на основі шаблонів хребтів і борозд на поверхні кінчиків пальців. Удосконалити первинну обробку для покращення контрастності зображення та зменшення помилок при вторинній обробці.

Наукова новизна полягає в наступному:

Розроблений спосіб є адаптивним для ідентифікації та підходить для різностороннього застосування. Використання ідентифікації відбитків пальців у блочній структурі збільшує правдивість та захищеність файлів у ланцюжку з використанням сканеру у будь-якому смартфоні чи за допомогою підключеного сканеру до комп'ютера.

Практична цінність полягає у підвищенні ефективності захисту інформації з використанням ідентифікації відбитків пальців для проектів на blockchain.

Апробація роботи. Основні положення і результати роботи будуть представлені та обговорюватимуться на науковій конференції магістрантів та аспірантів «Прикладна математика та комп'ютинг» ПМК-2018 (Київ, 22-24 березня 2018 р.).

Структура та обсяг роботи. Магістерська дисертація складається з вступу, п'яти розділів та висновків.

У вступі подано загальну характеристику роботи, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень, показано наукову новизну отриманих результатів і практичну цінність роботи.

У першому розділі розглянута предметна область дослідження обраної теми магістерської дисертації.

У другому розділі описується первинна обробка відбитків пальців

У третьому розділі описується спосіб обробки зображення

У четвертому розділі описується спосіб інтеграції методів ідентифікації у blockchain

У п'ятому розділі показується результат роботи із конкурентним порівнянням і її ефективність

У висновках представлені результати проведеної роботи.

Робота представлена на 82 аркушах, містить посилання на список використаних літературних джерел.

Ключові слова: біометрія, розпізнавання відбитків пальців, індексування, kd-дерева, автентифікація, ідентифікація

ABSTRACT

Relevance of the topic. Fingerprint is one of the most common biometric methods for data recognition through reliability, speed and performance. Imprints remain the same for a person's entire life. According to these advantages, the use of biometric fingerprints is increasingly being used by various commercial and government organizations. The fingerprint identification feature allows you to automatically and securely extract the minuses from input fingerprint images. However, the effectiveness of the minus removal algorithm depends on the quality of the input fingerprints. To ensure the reliability of the fingerprint authentication system, it is necessary to pre-process the image. This thesis describes the steps that are taken during the pre-processing of fingerprints, which improves the clarity of the spine structure and the bifurcation of the input fingerprints. After preprocessing, the minuses are displayed and stored in the database. Next, an online fingerprint authentication system is introduced, which uses the elementary indexing strategy. Fingerprint indexing is performed to identify and obtain a small subset of candidate data from the database. Experimental work shows that the integration of the online system of the pre-processing algorithm improves the overall response time.

The object of research is the method of improving the effectiveness of information security.

The subject of the study are methods for improving the effectiveness of information security by identifying fingerprints for projects on the blockchain.

The purpose and objectives of the study:

Create a way to effectively protect information for projects that use block structure using fingerprint identification at any stage of the recording of information.

Conduct experiments on the algorithms of the chosen method for identifying different types of fingerprints based on templates of ridges and furrows on the surface of the fingertips. Improve the primary processing to improve the contrast of the image and reduce the errors in the secondary processing.

Scientific novelty consist in the following: The developed method is adaptive for identification and suitable for versatile applications. Using fingerprint identification in a block structure increases the truth and security of files in a chain using a scanner on any smartphone or through a connected scanner to a computer.

The practical value is to increase the effectiveness of protection fingerprint identification information for projects on the blockchain.

Approbation of method. The main provisions and results of the work will be presented and discussed at the scientific conference of masters and postgraduates "Applied Mathematics and Computer", PMK-2018 (Kyiv, March 22-24, 2018).

Structure and amount of work. The master's dissertation consists of an introduction, five sections and conclusions.

The introduction gives a general description of the work, substantiates the relevance of the research direction, formulates the purpose and objectives of the research, shows the scientific novelty of the results obtained and the practical value of the work.

In the first section the subject area of the study of the chosen theme of the master's thesis is considered.

*The second sec*tion describes the primary processing of fingerprints.

The third section describes how to handle the image.

The fourth section describes how to integrate identification methods in the blockchain.

The fifth section shows the result of work with competitive comparison and its effectiveness.

The conclusions are the results of the work.

The work is presented on 82 pages, contains a link to the list of used literary sources.

Keywords: biometrics, fingerprint recognition, indexing, kd-trees, authentication, identification

РЕФЕРАТ

Актуальность темы. Отпечаток пальцев - один из самых распространенных биометрических способов для распознавания данных через надежность, скорость и производительность. Отпечатки остаются неизменными в течение всей жизни человека. Согласно этим преимуществ применения биометрических отпечатков пальцев все больше используется различными коммерческими и государственными организациями. Функция идентификации отпечатков пальцев позволяет автоматически и надежно извлекать минусы из входящих изображений отпечатков пальцев. Однако эффективность алгоритма извлечения минусов зависит от качества входных отпечатков пальцев. Чтобы обеспечить надежность работы системы проверки подлинности по отпечаткам пальцев, необходимо предварительно обрабатывать изображения. Этот тезис описывает шаги, происходящие во время предварительной обработки отпечатков пальцев, улучшает четкость структуры позвоночника и бифуркации входных отпечатков пальцев. После предварительной обработки минусы выводятся и сохраняются в базе данных. Далее внедрена онлайн-систему аутентификации отпечатков пальцев, в которой используется элементарная стратегия индексирования. Индексирования данных отпечатков пальцев осуществляется для идентификации и получения небольшой подмножества данных кандидатов из базы данных. Экспериментальные работы показывают, что интеграция онлайн-системы алгоритма предварительной обработки улучшает общее время отклика.

Объектом исследования является метод повышения эффективности защиты информации.

Предметом исследования являются методы повышения эффективности защиты информации с использованием идентификации отпечатков пальцев для проектов на blockchain**.**

Цель и задачи исследования: Создать образ эффективной защиты информации для проектов, которые используют блочную структуру с использованием идентификации отпечатков пальцев на любых этапах записи информации.

Провести эксперименты по алгоритмам выбранного способа по идентификации различных типов отпечатков пальцев на основе шаблонов хребтов и борозд на поверхности кончиков пальцев. Усовершенствовать первичную обработку для улучшения контрастности изображения и уменьшения ошибок при вторичной обработке.

Научная новизна заключается в следующем:

Разработанный способ является адаптивным для идентификации и подходит для разностороннего применения. Использование идентификации отпечатков пальцев в блочной структуре увеличивает правдивость и защищенность файлов в цепочке с использованием сканера в любом смартфоне или с помощью подключенного сканера к компьютеру.

Практическая ценность заключается в повышении эффективности защиты информации с использованием идентификации отпечатков пальцев для проектов на blockchain.

Апробация работы. Основные положения и результаты работы будут представлены и будут обсуждаться на научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2018 (Киев, 22-24 марта 2018).

Структура и объем работы. Магистерская диссертация состоит из введения, пяти глав и выводов.

Во введении представлена общая характеристика работы, обоснована актуальность направления исследований, сформулированы цели и задачи

исследований, показано научную новизну полученных результатов и практическую ценность работы.

В первой главе рассмотрена предметная область исследования выбранной темы магистерской диссертации.

Во втором разделе описывается первичная обработка отпечатков пальцев.

В третьем разделе описывается способ обработки изображения.

В четвертом разделе описывается способ интеграции методов идентификации в blockchain.

В пятом разделе показывается результат работы с конкурентным сравнением и ее эффективность.

В выводах представлены результаты проведенной работы.

Работа представлена на 82 листах, содержит ссылки на список использованных литературных источников.

Ключевые слова: биометрия, распознавание отпечатков пальцев, индексирования, kd-дерева, аутентификация, идентификация

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Блокчейн, тобто ланцюжок блоків транзакцій (англ. Blockchain, Block chain від block — блок, chain — ланцюг) — розподілена база даних, яка підтримує перелік записів, так званих блоків, що постійно зростає. База захищена від підробки та переробки. Кожен блок містить часову мітку та посилання на попередній блок хеш дерева.

Хеш дерева (tiger tree tashing, англ. Merkle tree) представляє собою особливу структуру даних, яка містить підсумкову інформацію про якийсь більший обсяг даних. Використовується для перевірки цілісності даних.

Біфуркація - термін походить від лат. bifurcus - «роздвоєний» і вживається в широкому сенсі для позначення всіляких якісних перебудов чи метаморфоз різних об'єктів при зміні параметрів, від яких вони залежать.

Кореляція - залежність між величинами (параметрами), що не має чіткого функціонального характеру.

Мінуси - ознаки вхідних зображень відбитків пальців.

Автентифікація - процедура встановлення належності користувачеві інформації в системі пред'явленого ним ідентифікатора.

Біометрія — сукупність автоматизованих методів і засобів ідентифікації людини, заснованих на її фізіологічній або поведінковій характеристиці.

ВСТУП

Відбиток пальців – один із найпоширеніших біометричних способів для розпізнавання даних через надійність, швидкість та продуктивність. Відбитки залишаються незмінними протягом усього життя особи. Відповідно до цих переваг застосування біометричних відбитків пальців все більше використовується різними комерційними та державними організаціями. Функція ідентифікації відбитків пальців дозволяє автоматично та надійно витягати мінуси (ознаки) з вхідних зображень відбитків пальців[1]. Проте ефективність алгоритму вилучення мінусів залежить від якості вхідних зображень відбитків пальців. Щоб забезпечити надійність роботи системи перевірки автентичності за відбитками пальців, необхідно попередньо обробляти зображення. Ця теза описує кроки, що відбуваються під час попередньої обробки відбитків пальців, що покращує чіткість структури хребта та біфуркації вхідних відбитків пальців. Після попередньої обробки мінуси виводяться та зберігаються в базі даних. Далі впроваджено онлайн-систему автентифікації відбитків пальців, в якій використовується елементарна стратегія індексування (сканування). Індексування даних відбитків пальців здійснюється для ідентифікації та отримання невеликої підмножини даних кандидатів з бази даних. Експериментальні роботи показують, що інтеграція онлайн-системи алгоритму попередньої обробки покращує загальний час відгуку. Тому доцільно додатково адаптувати методи і алгоритми попередньої обробки зображень при розпізнаванні відбитків пальців для підвищення ефективності реалізації.

1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ДОСЛІДЖЕНЬ БІОМЕТРІЇ ВІДБИТКІВ ПАЛЬЦІВ

1.1 Особливості біометрії відбитків пальців

Метод ідентифікації відбитків все більше стає популярним через його відносно видатні особливості:

- універсальность;
- стійкість;
- унікальність;
- точність;

- низька вартість розробки.

Ця техніка є популярною та надійною, в даний час є провідною біометричною технологією. Ідентифікація відбитків пальців є однією з найважливіших біометричних технологій, яка останнім часом привернула значну увагу. Відбиток є сукупністю шаблону хребтів і борозд на поверхні кінчика пальця. Унікальність відбитка пальців виключно визначається місцевими характеристиками хребта. Локальні характеристики хребта не рівномірно розподілені. Дві найвідоміші характеристики хребта, звані мініатюрами, - хребет кінця і хребетна біфуркація. Кінцевий хребет визначається як точка де хребет закінчується різко. Хребетна біфуркація визначається як точка, де знаходиться хребет вилки або розбігаються в хребти гілки. Якісний відбиток зазвичай містить близько 40-100 мініатюр. Проте на практиці через різницю в умовах враження, конфігурації гряди, стану шкіри (аномальних утворень епідермальних гребенів відбитків пальців, постнатальних міток, професійних знаків) значний відсоток має погану якість. Хребетні структури в неякісних відбитках зображень не завжди добре визначені і, отже, вони не можуть бути коректно виявлені. Це призводить до виникнення таких проблем, як значне число недостовірних мінусів, великий відсоток справжніх мініатюр може бути проігноровано, і можуть бути введені більші помилки в їх локалізації (положення та орієнтація). Тому через ці проблеми виконується попередня обробка вхідного зображення, а потім функція оброблюється. Первинна обробка [2] здійснюється за допомогою біометрії відбитків пальців, щоб покращити контрастність зображення та зменшити кількість помилок.

1.2 KD-дерева

KD дерева є корисною структурою даних для пошуку багатовимірного ключа. На рисунку 1.1. показано приклад KD-дерева, котре використовується в роботі по ідентифікації відбитків пальців для блокчейн проектів.

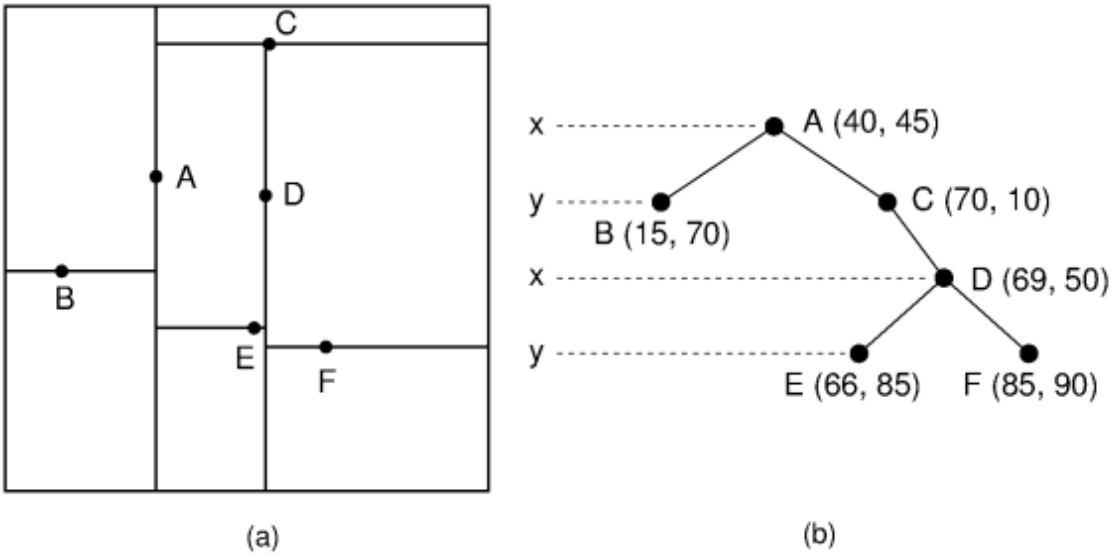


Рис. 1.1 - KD-дерево використаного у розробці

Дерево k-d - це бінарне дерево, в якому кожний вузол є k-мірною точкою. Кожен нелітинний вузол можна вважати таким, що неявно генерує розщеплюючу гіперплоскість, яка ділить простір на дві частини, відомі як пробіли. Ліворуч представлені піддерева цього вузла, а точки правої гіперплощі представлені правим піддеревом. Напрямок гіперплощі вибирається таким чином: кожен вузол у дереві асоціюється з одним із k-розмірів, причому гіперплоща перпендикулярна осі цієї величини. Так, наприклад, якщо для певного розщеплення вибрано вісь "x", всі лінії в піддереві з меншим значенням "x", ніж вузол, з'являться у лівому піддереві, і всі точки з більшим значенням "x" будуть в правому піддереві. У такому разі гіперплоскість буде встановлюватися за допомогою x-значення точки, а його нормаль буде одиницею x-осі. Операції на k-d деревах - Insert, Delete, FindMin, Search та Nearest Neighbor, описані нижче.

1.2.1 Операції

1.2.1.1 Insert (введення)

Вставка виконується за допомогою алгоритму 1. Вставки в KD-дереві схожі на будь-яке інше дерево пошуку. KD -дерево перетинається, щоб знайти відповідний лист, починаючи з кореня залежно від того, чи точка, яку потрібно вставити, знаходиться на "лівому" або "правому" боці, щоб вмістити новий запис. Запис вставляється як ліва або права дитина вузла, залежно від того, яка сторона площини розщеплення вузла містить новий вузол. У випадку дубліката запису вона не вставляється, та з’являється помилка.

Алгоритм 1. Введення

```
  Input: T: Tree node, P: Point to be inserted, l: level, D:
Tree Dimension Output:
  T: root node
  1: procedure Insert(T, P, l)
  2: if T is null then
  3: Create new node including point P
  4: else if T.data equals P then
  5: Notify duplicate entry
  6: else if T.data[l] > P[l] then
  7: T.lef t ←Insert(T.lef t, P,(l + 1) mod D)
  8: else
  9: T.right ← Insert(T.right, P,(l + 1) mod D)
  10: end if
  11: return T
  12: end procedure
```

1.2.1.2 Delete (видалення)

Щоб видалити точку з існуючого дерева KD, створюється набір всіх вузлів і листів від дітей цільового вузла та відтворюється ця частину дерева. Видалення виконується за допомогою алгоритму 2.

Алгоритм 2. Видалення

```
  Input: T: Tree node, P: Point to be Deleted, l: level, D:
Tree Dimension Output:
  T: root node
  1: procedure Delete(T, P, l)
  2: if T is null then
  3: Notify point not found.
  4: return T
  5: end if
  6: if T.data equals P then
  7: if T.right 6= null then
  8: T.data ←FindMin( T.right, l,(l + 1) mod D)
  9: T.right ←Delete(T.right, T.data,(l + 1) mod D)
  10: else if T.lef t 6= null then
  11: T.data ←FindMin(T.lef t, l,(l + 1) mod D)
  12: T.lef t ←Delete(T.lef t, T.data,(l + 1) mod D)
  13: else
  14: T ← null
  15: end if
```

```
16: else if T.data[l] > P[l] then
17: T.lef t ←Delete(T.lef t, P,(l + 1) mod D)
18: else
19: T.right ←Delete(T.right, P,(l + 1) mod D)
20: end if
21: return T
22: end procedure
```

1.2.1.3 FindMin (пошук мінімального)

FindMin допомагає знайти точку з найменшим значенням у параметрі dth. FindMin виконується з алгоритмом 3.

Алгоритм 3. Пошук мінімального

```
Input: T: Tree node, l: level, D: Tree Dimension
Output: R: Tree node
1: procedure FindMin(T, l)
2: if T is null then
3: Notify point not found.
4: return null
5: end if
6: if l equals D then . T splits on the dimension were searching
7: l ← (l + 1) mod D
8: if T.lef t == null then
9: R ← T
10: return R
11: else
12: return FindMin(T.lef t, l)
13: end if
14: else
15: R ←Minimum(FindMin(T.lef t, l), FindMin(T.right, l))
16: end if
17: return R
18: end procedure
```

1.2.1.4 Search (пошук)

Пошук виконується за алгоритмом 4.

Алгоритм 4. Пошук мінімального

```
Input: T: Tree node, P: Point of interest
Output: T: Tree node
```

```
1: procedure Search(T, P)
2: for all l such that T! = null do
3: if T.data equals P then
4: return null
5: else if T.data[l] > P[l] then
6: T ← T.lef t
7: else
8: T ← T.right
9: end if
10: l ← (l + 1) mod D
11: end for
12: return null
13: end procedure
```

1.2.1.5 RangeSearch (виявлення діапазону пошуку)

Виявлення діапазону пошуку виконується за алгоритмом 5.

Алгоритм 5. Виявлення діапазону пошуку

```
Input: T: Tree node, P1: Point Low , P2: Point High, l: level, D: Tree Dimension
Output: R: List of Node(s)
1: procedure RangeSearch(T, P1, P2, l)
2: if T is null then
3: return R
4: end if
5: if P1[l] <= T.data[l] then
6: RangeSearch(T.lef t, P1, P2,(l + 1) mod D)
7: end if
8: j ← 0
9: while P1[l] <= T.data[l] <= P2[l] do
10: increment j
11: end while
12: if j == l then
13: Add T to R
14: end if
15: if P2[l] > T.data[l] then
16: RangeSearch(T.right, P1, P2,(l + 1) mod D)
17: end if
18: end procedure
```

1.2.1.6 Nearest Neighbor (пошук найближчого сусіда)

Алгоритм пошуку найближчого сусіда (NN) має на меті знайти точку в дереві, що найближча до заданої точки. Цей пошук можна зробити ефективно за допомогою властивостей дерева, щоб швидко виключити великі частини пошукового простору.

Алгоритм 6. Пошук найближчого сусіда

```
Input: T: Tree node, P: Point to be inserted, l: level, BB: Bounding Box Rectangle,
D: Tree Dimension
Output: R: Nearest Neighbor node
1: procedure NearestNeighbor(T, P, l, BB)
2: if T is null or distance(P, BB) > BestDistance then
3: R ← T
4: return R . if this bounding box is too far, then do nothing
5: end if
6: dist ← distance(Q, T.data) . If this point is better than the best
7: if dist < BestDistance then
8: BestDistance ← dist
9: end if
10: l ← (l + 1) mod D
11: if P[l] < T.data[l] then
12: R ← NearestNeighbor(T.lef t, P, l, BB.T rimLef t(l, T.data))
13: R ←NearestNeighborT.right, P, l, BB.T rimRight(l, T.data))
14: else
15: R ←NearestNeighbor(T.right, P, l, BB.T rimRight(l, T.data))
16: R ←NearestNeighbor(T.lef t, P, l, BB.T rimLef t(l, T.data))
17: end if
18: end procedure
```

2. ПОПЕРЕДНЯ ОБРОБКА ВІДБИТКІВ

2.1 Визначення мініатюр на відбитках

Традиційно два відбитка пальця порівнювалися з використанням дискретних функцій, які називаються мінімуми. Ці особливості включають точки в області тертя пальця, де кінець гребенів (званий кінцем гребеня) або поділ (зване біфуркацією хребта). Як правило, на відбитку близько 100 мініатюр. Для пошуку і зіставлення відбитків пальців записуються координати і орієнтація гребеня в кожній точці мініатюри. На рисунку 2.1 показаний приклад двох типів мініатюр. Мініатюри відзначені на правильному зображенні, а хвости на маркери вказують в напрямку орієнтації мініатюр.

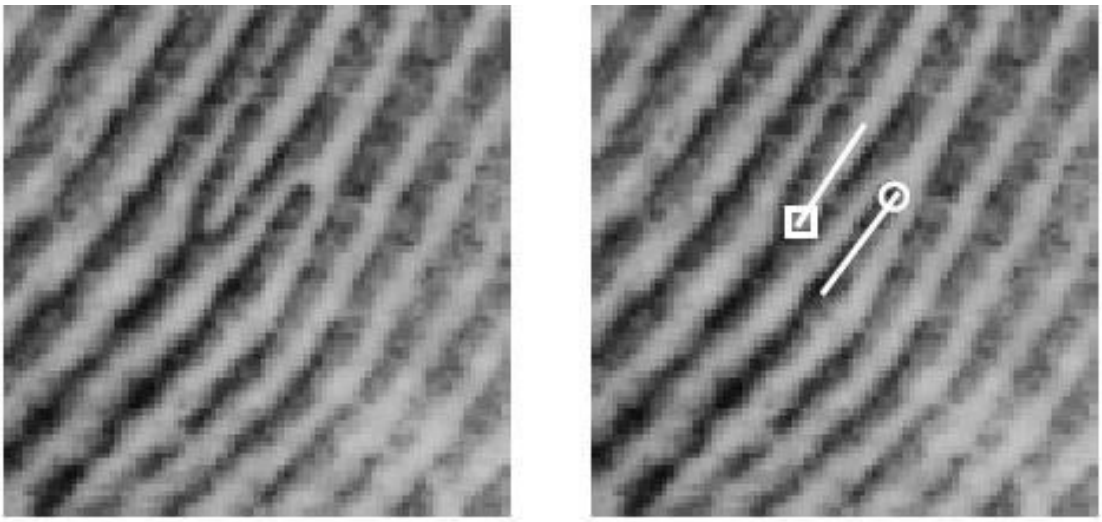


Рис. 2.1 - Мініатюри відбику пальця

Розташування кожної мініатюри представлено координатним місцем розташування в зображенні відбитка пальця. Різні системи AFIS представляють це місце розташування по-різному. Стандарт ANSI / NIST [3] визначає одиниці відстані в 0,01 мм від початку координат в лівому нижньому кутку зображення. Наприклад, зображення розміром 500 x 600 пікселів, скановане на 19,69 пікселів на міліметр (ppm m) має розміри 25,39 x 30,47 мм, що в стандартних одиницях 0,01 мм. Таким чином, координата пікселя (200, 192) буде представлена в стандартних одиницях, де координата Y вимірюється знизу зображення вгору.

Орієнтація мінімумів представлена в градусах, з нульовими градусами, що вказують горизонтально і вправо, і збільшуючи градуси, що йдуть проти годинникової стрілки. Орієнтація закінчення гребеня визначається шляхом вимірювання кута між горизонтальною віссю і лінією, що починається в точці мінімуму, і що проходить через середину хребта. Орієнтація біфуркації визначається шляхом вимірювання кута між горизонтальною віссю і лінією, що починається в точці мінімуму, і що проходить через середину проміжної долини між роздвоєними гребенями.

Кожен символ мініатюри складається з кола або квадрата, що позначає місце розташування мініатюрної точки, а лінія або хвіст, що йдуть від кола або квадрата, проектуються вздовж хребта ковзанів або долини біфуркацій.

2.2 Приховані відбитки пальців

Додатково до відбитків, також існує невелика кількість відбитків пальців. Це відбитки пальців, зняті в місцях злочину, які можуть бути використані як доказ у вирішенні кримінальних справ. На відміну від відбитків, які були захоплені в контрольованому середовищі з вираженою метою ідентифікації, відбитки пальців на місці злочину, за своєю природою, випадково залишилися позаду. Вони часто невидимі для очей без будь-якої хімічної обробки або пилу. Саме з цієї причини їх традиційно називають латентними відбитками пальців. Як і слід було очікувати, склад і якість прихованих відбитків пальців значно відрізняються від відбитків. Як правило, в латентному стані присутній тільки частини пальця, поверхня, на яку була відображена латентність, непередбачувана, а чіткість деталей шкіри тертя часто розмита або закупорена. Все це призводить до відбитків пальців значно меншої якості, ніж типові відбитки, як показано на рисунку 2.2.

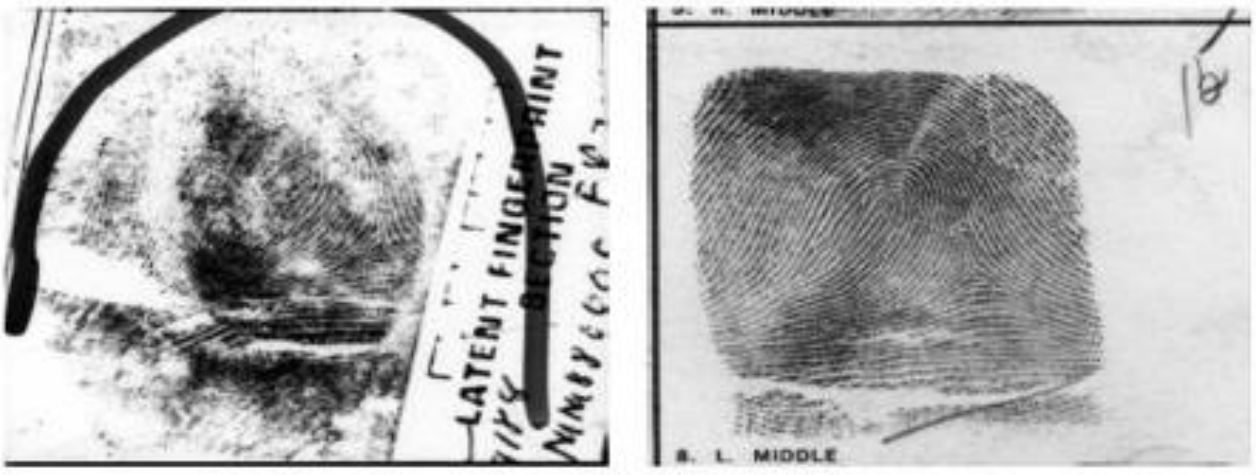


Рис 2.2 - Приховані відбитки пальців

Через погані умови прихованих відбитків пальців сьогоднішня технологія AFIS працює не досконало при поданні прихованого зображення відбитка пальця. Автоматизовані паспортні системи надзвичайно складно точно класифікувати приховані відбитки пальців і надійно виділити мінімуми на зображенні. Отже, фахівці з відбитками пальців людини, звані латентними екзаменаторами, повинні аналізувати і вручну маркувати кожен прихований відбиток пальця при підготовці до зіставлення. Це виснажливе і трудомістке завдання. Для підтримки обробки прихованих відбитків пальців FBI і NIST спільно розробили спеціалізовану робочу станцію під назвою Universal Latent Workstation (ULW). Ця робоча станція була розроблена для допомоги прихованого екзаменатора в підготовці прихованого відбитка пальця для пошуку. Крім того, робоча станція забезпечує інтероперабельність між різними системами AFIS, функціонуючи як незалежний від постачальника інтерфейсний інтерфейс. Ці два аспекти ULW в значній мірі сприяють поліпшенню стану латентної ідентифікації відбитків пальців і правоохоронної діяльності.

2.3 Процес виявлення мініатюр

Слід зазначити, що в алгоритмах виявлення мінімумів були розроблені алгоритми і параметри програмного забезпечення для оптимальної обробки

зображень, відсканованих зі швидкістю 12700 пікселів на міліметр і квантованих до 256 рівнів сірого.

Програма для виявлення мініатюр в зображенні відбитка пальця. Програмне забезпечення було розроблено модульно, так що кожен з кроків в основному виконується однією підпрограмою. Це дозволяє застосовувати інші альтернативні підходи і підставляти їх в процес, і загальний вплив на продуктивність може бути оцінений. Для підтримки багатьох необхідних робочих параметрів для запису розмірів, допусків і граничних значень використовується одна глобальна структура управління.

2.4 Обробка файлу скану відбитка пальців

Mindtct вводить зображення відбитка пальця і автоматично виявляє мініатюри на відбитку. Алгоритми і параметри були розроблені і встановлені для зображень, відсканованих на 19,69 p/mm і квантованих до 256 рівнів сірого. Додаток може читати файли в форматах ANSI / NIST, WSQ, JPEGB, JPEGL і IHEAD. [4] У форматованих файлах ANSI / NIST він шукає файлову структуру для запису відбитків пальців у відтінках сірого. В даний час обробляється тільки перший запис відбитка пальця в відтінках сірого в файлі ANSI / NIST, але додаток може бути змінено для обробки всіх відбитків пальців у відтінках сірого в файлі ANSI / NIST. У Mindtct є опція, яка дозволить їй покращувати зображення з дуже низькою контрастністю. Mindtct буде оцінювати гістограму вхідного зображення. Якщо зображення з дуже низькою контрасністю, воно автоматично покращюється для поліпшення контрастності.

2.5 Створення мап якості зоображень

Оскільки якість зображення відбитка пальця може варіюватися, особливо в разі прихованих відбитків пальців, важливо мати можливість аналізувати зображення і визначати області, які погіршуються і можуть викликати проблеми.

Можна виміряти кілька характеристик, які призначені для передачі інформації про якість локалізованих областей зображення. Вони включають в себе визначення спрямованого потоку гребенів в зображенні і детектування областей з низьким контрастом, низьким потоком гребеня і високою кривизною. Ці три останніх умови являють собою нестійкі області зображення, де виявлення дрібниць ненадійно, і разом вони можуть використовуватися для подання рівнів якості зображення.

2.5.1 Мапа напрямків

Одним з фундаментальних кроків в цьому процесі виявлення мініатюр є отримання орієнтовної карти потоку хребтів або карти напрямків. Метою цієї карти є уявлення областей зображення з достатньою структурою гребеня. Добре сформовані і чітко видимі хребти необхідні для надійного виявлення точок закінчення хребта і біфуркації. Крім того, мапа напрямків записує загальну орієнтацію гребенів, що проходять через зображення.

Для локального аналізу відбитка пальця зображення ділиться на сітку блоків. Всі пікселі всередині блоку отримують однакові результати. Тому в цьому разі всім пікселям в блоці буде призначено один і той самий напрямок потоку хребта. При використанні блокового підходу необхідно враховувати кілька міркувань. По-перше, необхідно визначити, скільки місцевої інформації потрібно для надійного отримання бажаної ознаки. Ця область називається вікном. Після чого характеристика, виміряна в вікні, присвоюється кожному пікселю в блоці. Зазвичай бажано обмінюватися даними, що використовуються для обчислення результатів, що привласнюються сусіднім блокам. Таким чином, деякі з зображень, які сприяли результатим одного блоку, також включені в результати сусідніх блоків. Це допомагає звести до мінімуму розрив у значеннях блоку при переході кордону з одного блоку в сусідній. Це згладжування може бути реалізовано з використанням системи, де блок менше, ніж його сусіднє вікно, і вікна перекриваються від одного блоку до іншого. Це

показано на рисунку 2.3.

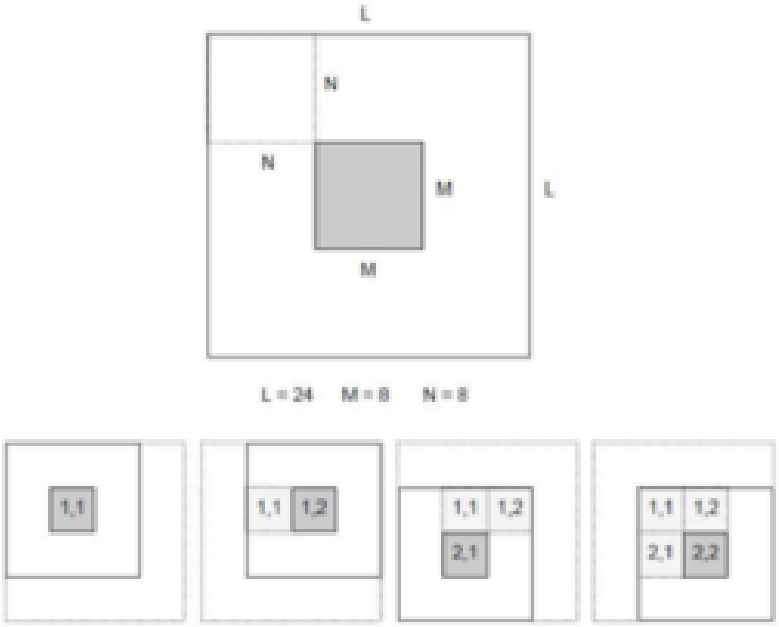


Рис. 2.3 - Мапа напрямків

Велика рамка у верхній частині малюнка зображує вікно, що оточує менший блок (в сірому кольорі). Припускаючи, що сусідні блоки суміжні і не перекриваються, цей сценарій визначається трьома параметрами: розміром вікна L, розміром блоку M і зміщенням блоку від початку координат N. У глобальній структурі управління lfsparmsV2 ці параметри визначаються як MAPWINDOWSIZEV2 = 24, MAPBLOCKSIZEV2 = 8 і APWINDOWOFFSETV2 = 8 відповідно. В результаті зображення ділиться на сітку з 8x8 піксельних блоків, причому кожному блоку присвоюється результат з більшого навколишнього 24x24 пиксельного вікна, а область для вікон сусідніх блоків перекривається до 2/3. Нижній рядок кадрів на рисунку 2.3 ілюструє, як це працює на практиці. Позначаючи адресу блоку по його індексам (рядок, стовпець), лівий фрейм показує перший блок (1,1), який обчислюється. Наступний кадр переходить до наступного суміжному блоку справа, блок (1,2). Відповідно, вікно зсувається на 8 пікселів, а новий блок отримує свої результати. Використовуються дві копії зображення. Кожне вікно працює з вихідними даними зображення, а результати блоків записуються в окремий вихідний файл. Третій кадр на малюнку зображує конфігурацію вікна для блоку (2,1), а четвертий кадр показує його праве

сусідство.

Повинно бути визначено, як поводитися з контурами зображення. Розміри зображення, ймовірно, не будуть парними кратними блокам, а вікна, навколишні блоки по периметру зображення, можуть виходити від зображення. Програмний продукт зображення доповнюється краєм середніх сірих пікселів (встановлюється на інтенсивність 128). Цей край досить великий, щоб утримувати вікна периметра в зображенні. Обробка часткових блоків також враховується в правій і нижній частині зображення. З огляду на вищезазначений підхід для обчислення результатів блоку з перекриваємим вікном, можна описати методику, яка використовується для визначення напрямку потоку хребта в зображенні. Для кожного блоку на зображенні навколишнє вікно повертається поступово, і за будь-якої орієнтації виконується аналіз дискретного перетворення Фур'є (ДПФ). У верхньому лівому полі на малюнку зображено вікно зі своїми рядами, поверненими на 90 градусів проти годинникової стрілки, щоб вони були вирівняні по вертикалі.

Параметр NUMDIRECTIONS в глобальній структуру управління lfsparsV2 задає кількість аналізованих орієнтацій в півколі. Цей параметр встановлений на 16, створюючи приріст кута 11,2 градусів між кожною орієнтацією. Ці орієнтації зображені на колі. Нижній рядок на малюнку ілюструє інкрементне обертання рядків вікон при кожній конкретній орієнтації. При визначенні напрямку потоку хребта для блоку аналізується кожна його орієнтація вікна. В межах орієнтації пікселі вздовж піксельних ряду. 16 орієнтацій виробляють 16 векторів сум рядків. Кожен вектор сум рядків встановлений з 4 частотними коливаннями. Верхня форма хвилі на малюнку має один період, що проходить по всій довжині вектора. Дискретні значення для синусоїдальних і косинусоїдальніх функцій на 4 різних частотах обчислюються для кожної одиниці вздовж вектора. Суми рядків у векторі потім множаться на їх відповідні дискретні значення синуса, а результати накопичуються і групуються. Те ж обчислення виконується між сумами рядків у векторі і їх

відповідними дискретними косінусовимі значеннями. Потім квадратний синусоїдальний компонент додається до квадрату косинусного компонента, створюючи коефіцієнт резонансу.

2.5.2 Мапа характеристик

Інша частина зображення відбитка пальця, яка проблематична, коли справа доходить до виявлення дрібниць, які знаходяться в областях з високою кривизною. Мапа з високою кривою відображає блоки, які знаходяться в областях з високою кривизною відбитка пальця. Використовуються два різні способи. Перший, званий завихрення, вимірює кумулятивну змінну напрямку потоку хребта навколо всіх сусідів блоку. Друга кривизна вимірює найбільшу зміну напрямку між потоком хребтів блоків і потоком гребеня кожного з його сусідів. Подробиці вказані у вихідному коді. У разі виявлення дрібниць в цих блоках їх якість зменшується, тому що вони були виявлені в менш надійних частинах зображення. Мапа характеристик показана на рисунку 2.4.

Рис. 2.4 - Мапа характеристик відбитка

2.6 Бінаризація зображення

Алгоритм виявлення дрібниць в цій системі призначений для роботи на двурівнемому зображенні, де чорні пікселі є хребтами, а білі пікселі представляють долини в шкірі тертя пальця. Щоб створити це двійкове зображення, кожен піксель в чорному зображенні повинен бути проаналізований, щоб визначити, чи повинен йому

призначатися чорний або білий піксель. Цей процес називається бінаризація зображення. Пікселю присвоюється двійкове значення, засноване на напрямку потоку хребта, пов'язане з блоком, в якому знаходиться піксель. Якщо не було виявленого потоку гребеня для блоку поточного пікселя, тоді піксель буде встановлений автоматично білим. Напрямок потоку хребта зображений на рисунку 2.5

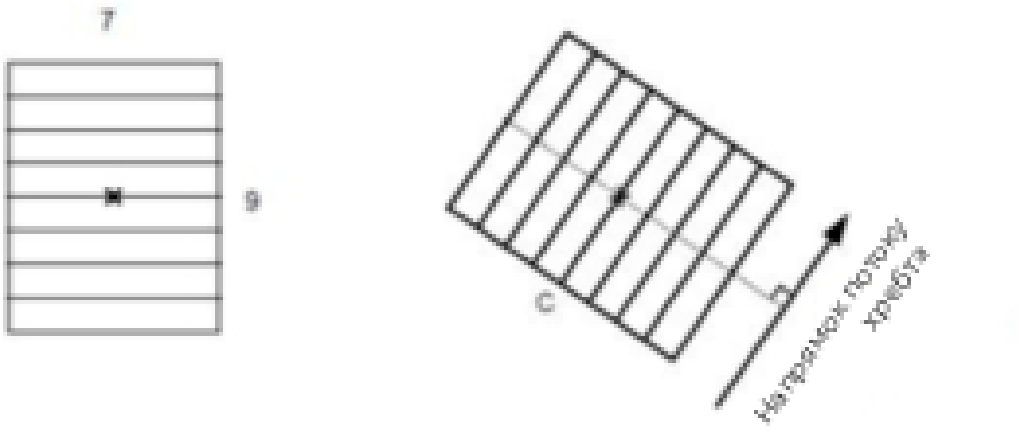


Рис. 2.5 - Напрямок потоку хребта

Поворотна сітка використовується для бінаризації зображення відбитка пальця. Ця сітка визначена в глобальну структуру управління, lfsparmsV2, з шириною стовпця (DIRBINGRIDW), встановленої в 7 пікселів, і висотою рядка (DIRBINGRIDH), встановленому на 9 пікселів. В центрі сітка повертається [5] так, що її ряди фіксуються паралельно напрямку потоку локального гребеня. Інтенсивність пікселів в відтінках сірого накопичуються уздовж кожного поверхненого рядку в сітці, утворюючи вектор сум рядків. Бінарне значення, яке має бути призначене центральному пікселю, визначається шляхом множення суми центрального рядка на кількість рядків у сітці і порівнянню цього значення з накопиченою інтенсивністю відтінків сірого в межах всієї сітки. Якщо сума помноженого центрального рядка менша, ніж загальна інтенсивності сітки, то центральний піксель встановлюється на чорний; в іншому випадку він буде білим.

Рис. 2.6. - Напрямок потоку хребта

Результати бінаризації показані на рисунку 2.6. Початкове зображення у відтінках сірого знаходиться зліва, а результати його бінарзиції - справа. Слід зазначити, що крок бінаризації має вирішальне значення для успішного виявлення мініатюр в цьому підході. Результати бінаризації повинні бути надійними з точки зору ефективної взаємодії з різними ступенями якості зображення і надійними з точки зору точності побудови хребтів і долин. Це складні і часом суперечливі моменти. Бажано зберегти якомога більше інформації про зображення і структури хребта/долин, щоб мініатюри не були пропущені.

2.7 Повторне виявлення мініатюр на відбитку

Цей крок методично сканує двійкове зображення відбитка пальця, ідентифікуючи локалізовані шаблони пікселів, які вказують на закінчення або поділ гребеня. Зразки дуже компактні, як показано на рисунку 2.7. Лівий шаблон містить шість двікової пікселів в конфігурації 2x3. Цей малюнок може являти собою кінець чорного хребта, котрий знаходиться праворуч від малюнка. Така ж схема буде вірною для наступного шаблону 2x4. Єдина відмінність між цим шаблоном і першим полягає в тому, що друга піксельна пара повторюється. Фактично, це вірно для всіх зображених малюнків. Це «сімейство» патернів закінчення хребта може бути представлено правим шаблоном, де середня пара

пікселів (позначається *). Вона може повторюватися один або кілька разів.

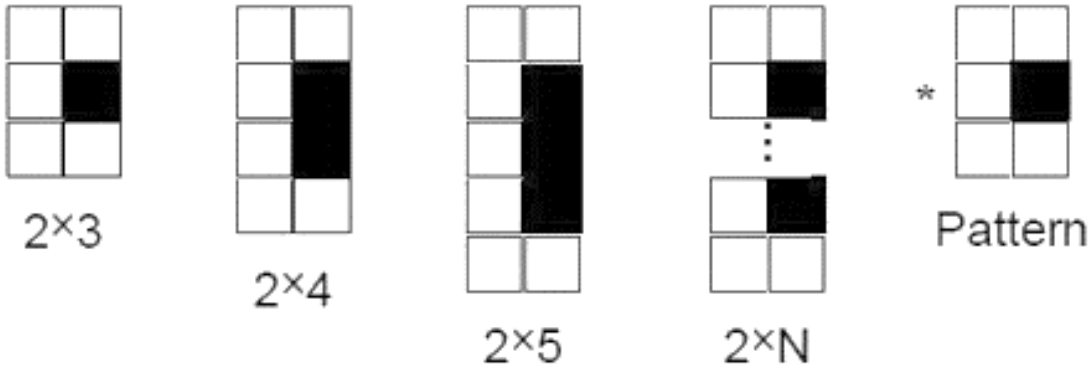


Рис. 2.7 - Локалізовані піксельні шаблони

Кінці хребтів виявляються в двійковому зображенні шляхом сканування послідовних пар пікселів на зображенні, які шукають послідовності, відповідні цим шаблонам. Сканування патернів виконується як по вертикалі, так і по горизонталі. Зразок налаштований для вертикального сканування, оскільки пари пікселів укладаються один на одного один над одним. Щоб провести горизонтальне сканування, пари пікселів будуть повернені на 90 градусів за годинниковою стрілкою і переміщенні назад послідовно вліво-вправо. Використовуючи вищенаведене уявлення, ряд мініатюрних шаблонів використовується для виявлення точок-кандидатів в мініатюрних точках в зображенні відбкитку. Ці малюнки показані на рисунку 2.8. Вторинний атрибут появи / зникнення присвоюється кожному шаблону. Це визначає напрямок, з якого гребінь або долина формуються в візерунок. Всі послідовності пар пікселів, що відповідають цим паттернам, у міру того, як зображення сканується як по вертикалі, так і по горизонталі, формують перелік кандидатів в мініатюрні точки.

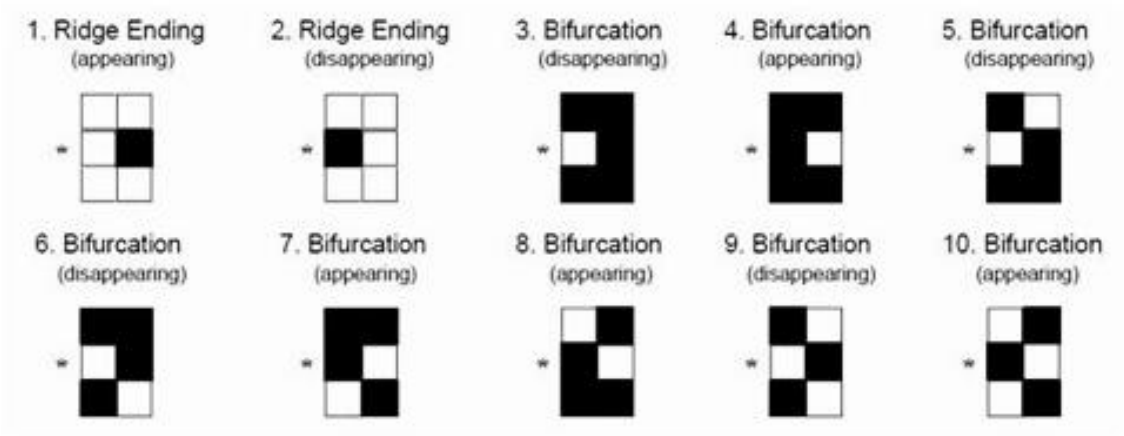


Рис. 2.8 - Моделі біфуркації хребта

2.8 Видалення помилкових мініатюр

Використовуючи схеми, наведені на рисунку 2.9, точками кандитатів є лише шість пікселів. Це полегшує особливо жадібну схему виявлення, яка мінімізує ймовірність відсутності справжніх мінімумів; проте багато помилкових мінусів включено до списку кандидатів. Через це великі зусилля витрачаються на видалення помилкових мінусів. Ці кроки включають видалення «островів», «озер», «дірок», мінусів у регіонах з низькою якістю зображення. Короткий опис кожного з цих кроків надано в тому порядку, в якому вони виконуються.

2.8.1 Видалення островків та маркувальних плям

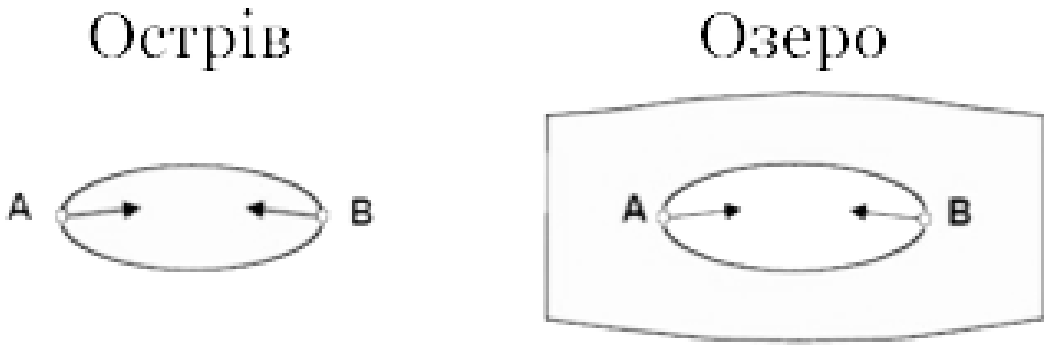


Рис. 2.9 - Ідентифікації та видалення островків та озер

Алготим 1. Видалення

```
1: procedure Remove Islands and Lakes (A, B)
```

```
2: if distance(A, B) <= 18pixels then
3: if directionAngle(A, B) >= 123.75deg then
4: if onLoop(A) AND onLoop(B) then
5: if loopLength <= 80pixels then
6: remove(A, B)
7: end if
8: end if
9: end if
10:end if
11:fillLoop()
12: end procedure
```

На цьому етапі ідентифікуються та видаляються фрагменти кінця хребта та маркувальні плями (острови) разом з внутрішніми порожнинами в хребтах (озерах). Ці характеристики дещо більші, ніж розміри пори в шкірі тертя, і вони часто еліптичні за формою; отже, вони, як правило, матимуть пару точок кандидатів, виявлених на протилежних кінцях. Ілюстрація цих типів функцій показана на рисунку 2.9. Включені в нижній частині малюнка критерії, які використовуються для виявлення островів та озер. Пара мінусів повинна бути в межах 16 пікселів (MAXRMTESTDISTV2) [6] один до одного. Якщо це так, то напрямки двох мінусів повинні бути майже напроти (123.75 градусів) один до одного. Потім обидва мінуси повинні знаходитися на краю одного циклу, а периметр циклу повинен бути 60 пікселів (MAXHALFLOOPV2). Якщо всі ці критерії є правильними, то пари кандидатів заносяться у перелік.

2.8.2 Видалення дірок

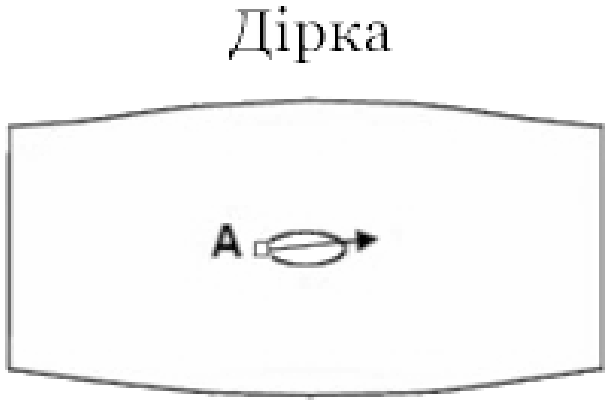


Рис. 2.10 - Видалення дірок

Алгоритм 2. Видалення дірок

```
1:  procedure Remove Holes (A, B)
2:      if  onLoop(A) AND onLoop(B)
3:      if loopLength <= 15pixels
4:      remove(A)
5:      end if
6:        end if
7:  end procedure
```

Отвір визначається аналогічно острову чи озеру і цикл має лише одну точку мінуса. Критерії для видалення отвору показані на рисунку 2.10. Якщо кандидат мінімума точки лежить на краю петлі з довжиною периметра 15 пікселів (SMALLLOOPLEN), то точка видаляється з списку кандидатів.

2.8.3 Вилучення вказівок на недійсний блок

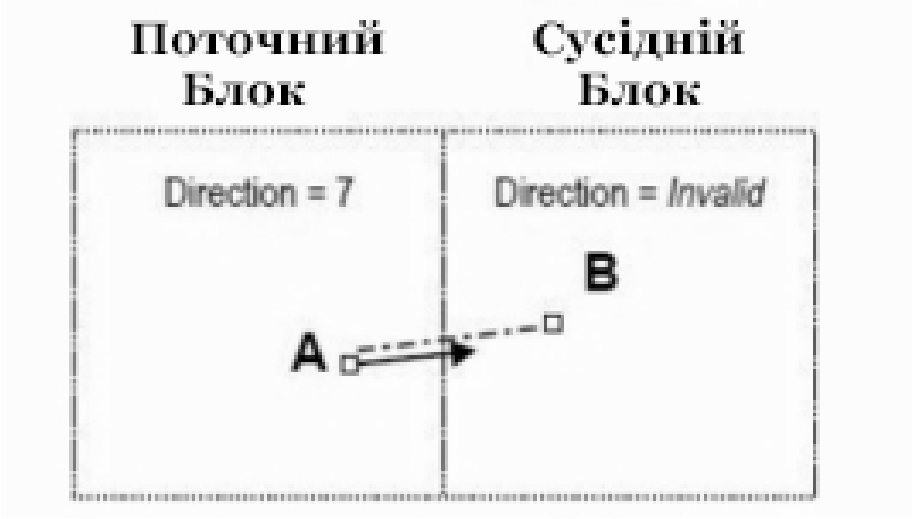


Рис. 2.11 - Видалення вказівки на недійсний блок

Алгоритм 3. Видалення вказівок на недійсний блок

```
1: procedure Remove Pointing to Invalid Block (A, B)
2: B = translate(A, 4pixels, direction(A))
3: D = direction(block(B))
4: if D is invalid then
5: remove(A)
6: end if
7: end procedure
```

Цей крок ідентифікує і видаляє кандидатські мінуси, розташовані біля блоків, які не містять виявленого потоку хребта. Ці блоки називається напрямком потоку хребта та представляє низькоякісні ділянки на зображенні відбитків пальців.

Цей крок показаний на рисунку 2.11. Точка мінуса перекладається 4 пікселями (TRANSDIRPIXV2) у напрямку, на який вказує мініатюра. Якщо переведена точка лежить у блоці з невірним напрямком потоку хребта, то вихідний мінімум видаляється зі списку.

2.8.4 Видалення даних біля недійсних блоків

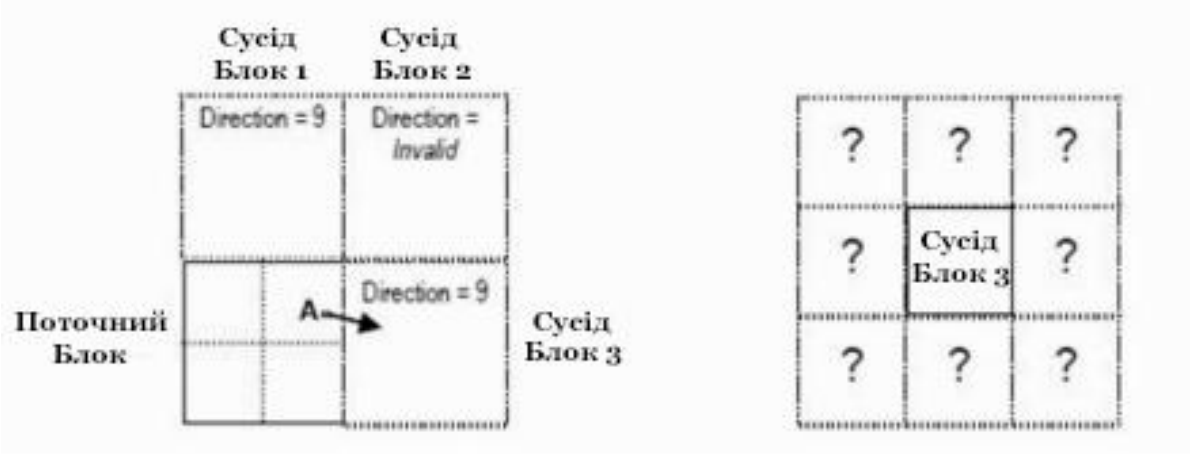


Рис. 2.12 - Видалення данних біля недійсних блоків

Алгоритм 4. Видалення данних біля недійсних блоків

```
1: procedure Remove Near Invalid Blocks (A)
2: Nbrs = blockN eighbors(A)
3: InvN brs = invalidDirections(Nbrs)
4: while Ni in InvN brs do
5: NiNbs = neighbors(Ni)
6: Ci = countV alidDirections(NiNbrs)
7: end while
8: if Ci < 7 then
9: remove(A)
10:end if
11: end procedure
```

В цьому алгоритмі оцінюється близькість кандидата мінімума до ряду навколишніх блоків з недійсним напрямком хребта. З огляду на точку мінуса, блоки, достатньо близькі до мінімумів (деталі залишаються до вихідного коду), а також безпосередньо сусідні з блоком, в якому розташований мінімум. Тестуються по черзі. Якщо один із сусідніх блоків має недійсний напрямок потоку хребта, то перевіряють його навколишні 8 сусідів. Підраховується кількість оточуючих блоків з діючим напрямком потоку хребта, а якщо кількість дійсних блоків менше 7 (RMVALIDNBRMIN), то вихідна точка мінімума буде видалена з списку кандидатів. Рисунок 2.12 іллюструє цей крок.

2.8.5 **Видалення або відрегулювавання бічних мініатюр**

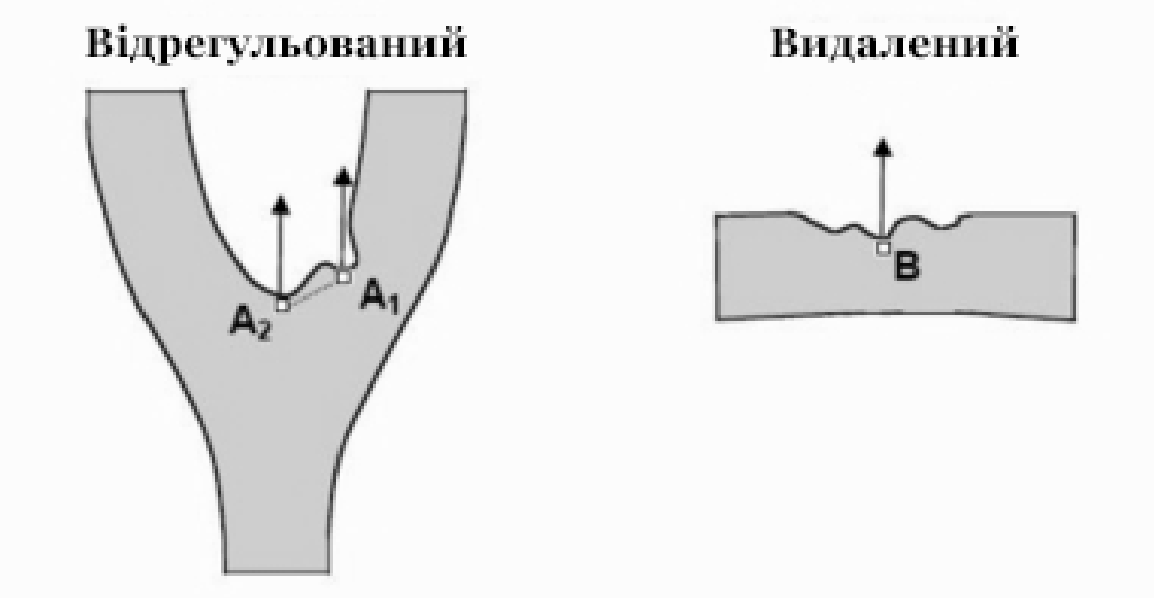


Рис. 2.13 - Видалення та регулювання бічних мініатюр

Алгоритм 5. Видалення або регулювання бічних мініатюр

```
1: procedure Remove or Adjust Side Minutiae (A)
2: P ts = traceContours(A, 7pixels)
3: RP ts = rotateP ointsV ertical(P ts, direction(A))
4: (M inY s, M axY s) = minM axY s(RP ts)
5: if M inY s == 1 then
6: Adjust(A, P ts[M inY 1])
7: else if (M inY s, M axY s) == (M inY 1, M axY 1) then
8: M inY = pointAtM inY (RP ts, M inY s)
9: Adjust(A, P ts[M inY ])
10:else
11:revove(A)
12:end if
13: end procedure
```

Цей крок виконує дві цілі. По-перше, потрібно точно настроїти позицію точок мінімумів, щоб вони були більш симетрично розташовані на кінці хребта або долини. У процесі може бути визначено, що немає чіткої симетричної форми для контуру, на якому лежить кандидат мінімума. Це часто відбувається з точками, виявленими уздовж хребта або долини замість хребта або кінця долини. У цьому випадку, неправильна точка мінімума вилучається. На рисунку 2.13, ілюстрація зліва зображує коригування точки мінуса з точки A1 до A2. На малюнку праворуч зображено зняття бокової точки B. Щоб це зробити,

починаючи з точки кандидата мінімума, край контура або долини простежується праворуч і ліворуч на 7 пікселів (SIDEHALFCONTOUR), створюючи список з 15 точок контура. Координати цих точок контуру обертаються таким чином, що напрямок кандидата мінімума вказує на вертикальний. Потім координати аналізуються для визначення числа та послідовності відносних максимумів та мінімумів у поворотних координатах.

Якщо існує лише одна мінімальна кількість у координат, передбачається, що точка мінімуму буде лежати в нижній частині обертованого контуру у формі кулі, а кандидат-мінімум буде перенесено на відповідність цій позиції у вихідному зображенні. Якщо існує декілька мінімальних у координат, то повинна існувати певна послідовність мінімумів-максимумів-мінімумів, в цьому випадку кандидат мінімума переноситься до точки. В оригінальному зображенні, що відповідає мінімальним у координатам. Знову ж таки, вважається, що це нижня частина відносно чашеобразного поворотного контуру. Якщо існує декілька мінімальних координат і не існує точних мінімумів-максимумів-мінімумів по оберненому контуру, то точка мінімуму визначається як лежата уздовж хребта або долини, і вона вилучається із списку кандидатів.

2.8.6 Видалення гачків

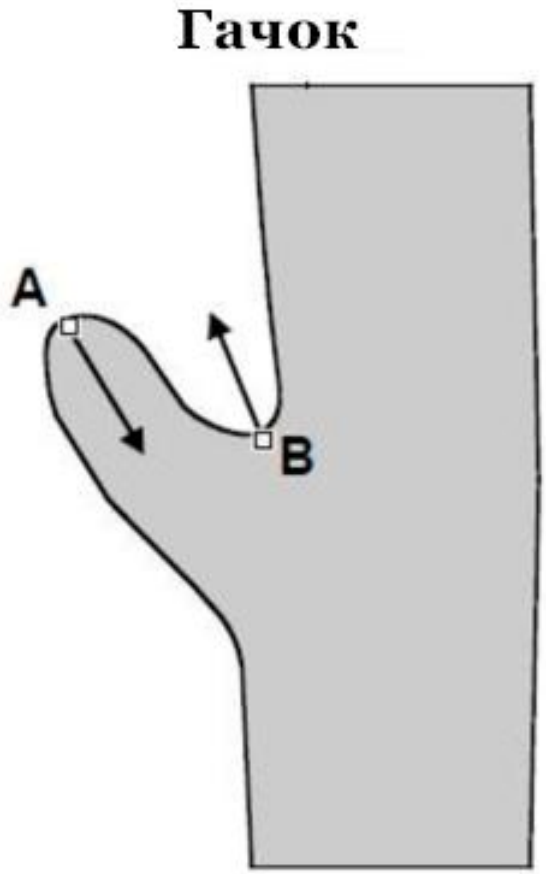


Рис. 2.14 - Видалення гачків

Алгоритм 6. Видалення гачків

Гачок - це шип або шпора, що виступає з боків хребта або долини. Приклад наведено на рисунку 2.14. Ця функція зазвичай має дві мінуси протилежного типу, один на невеликій частині хребта, а інший - у невеликій долині, що відносно близько один до одного. Ці дві точки мають бути в межах 16 пікселів (MAXRMTESTDISTV2) один з одним, їхні напрямки мають бути майже протилежними (123.75 градусів), вони повинні бути протилежного типу, і вони повинні лежати на тому ж краю долини в межах 30 контурних пікселів (MAXHOOKLENV2) [6] один від одного. Якщо все це вірно, то дві точки за хвилину видаляються з списку кандидатів

2.8.7 Видалення перекриттів

Рис. 2.15 - Видалення перекриттів

Алгоритм 7. Видалення перекриттів

```
1: procedure Remove Overlaps (A, B)
2: if distance(A, B) <= 8pixels then
3: if directionAngle(A, B) >= 123.75deg then
4: if type(A) == type(B) then
5: J=joinDirections(A, B)
6: if directionAngle(180deg – A, J) <= 90deg then
7: remove(A, B)
8: else if distance(A, B) <= 18pixels AND f reeP ath(A, B) then
9: remove(A, B)
10:end if
11:end if
12:end if
13:end if
14: end procedure
```

На цьому кроці перекриття є розривом у хребті або долині. Розрив в хребті призводить до виявлення 2 помилкових хребтів, а розрив в долині викликає 2 помилкових біфуркацій. Критерії виявлення перекриття зображено на рисунку 2.15. Дві точки повинні бути в межах 8 пікселів (MAXOVERLAPDIST) один до одної, а їхні напрямки повинні бути майже напроти (123.75 градусів) [7]. Якщо це так, то розраховується напрям лінії, що з'єднує дві точки. Якщо різниця близько 90 градусів, то дві мінуси вилучаються зі списку кандидатів. В іншому випадку, якщо мініатюри знаходяться в межах 6 пікселів (MAXOVERLAPJOINDIST) один від одного і немає перехідних значень пікселів вздовж лінії приєднання, то точки вилучаються з списку кандидатів.

2.8.8 Видалення занадто широких мінімумів

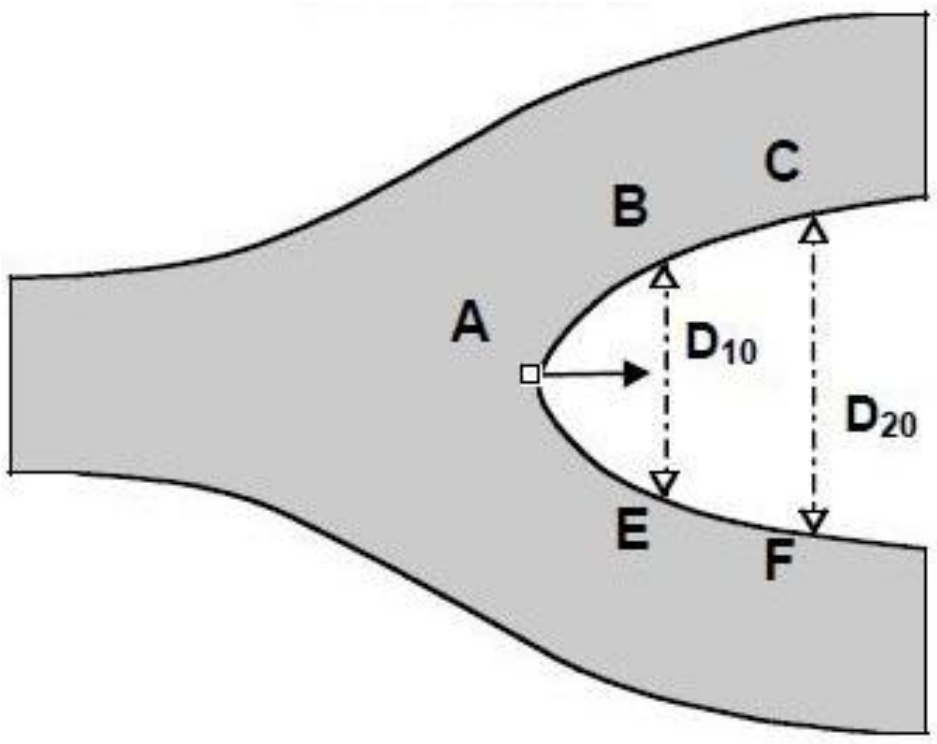


Рис. 2.16 - Видалення занадто широких мінімумів

Алгоритм 8. Видалення занадто широких мінімумів

```
1: procedure Removal of too wide minutiae (A, B)
2: Pts1 = traceContour(A, 20 pixels)
3: Pts2 = traceContour(A, -20 pixels)
4: B= Pts1(10); C = Pts1(20)
5: E = Pts2(10); F = Pts2(20)
6: D10 = distance(B, E)
7: D20 = distance(C, F)
8: if D20/D10 > 2.0 then
9: remove(A)
10:end if
11: end procedure
```

Наступні два кроки ідентифікують помилкові мінуси, що лежать на неправильно сформованих хребтах та долинах. Узагальнений хребетний кінець складається з Y-подібної долини, що огортає чорний стержень. Інверсія справедлива для узагальненої біфуркації. Просте тестування застосовується для оцінки якості Y-форми. Цей крок визначає, чи структура, що охоплює хребет або долину, закінчується відносно Y, і не дуже широка. На рисунок 2.16 наведені приклади критеріїв. Край хребта або долини простежується ліворуч та праворуч 20 пікселів (MALFORMATIONSTEPS2), що виробляють 2 списки точок контуру. На кожному контурі зберігаються координати в індексі пікселів 10 (B, E) і в

піксельному індексі 20 (C, F). Відстань між пікселями за індексом 10 (MALFORMATION- STEPS1) обчислюється так, як це відстань між пікселями в індексі 20. Потім розраховується співвідношення цих двох відстаней (D20 / D10), і якщо співвідношення є більшим за 2,0 (MINMALFORMATIONRATIO) , тоді пункт "мінімум" видаляється з списку каналів. Слід зазначити, що на підставі цих критеріїв біфуркація на ілюстрації не буде вилучена.

2.8.9 Видалення занадто вузьких мінімумів

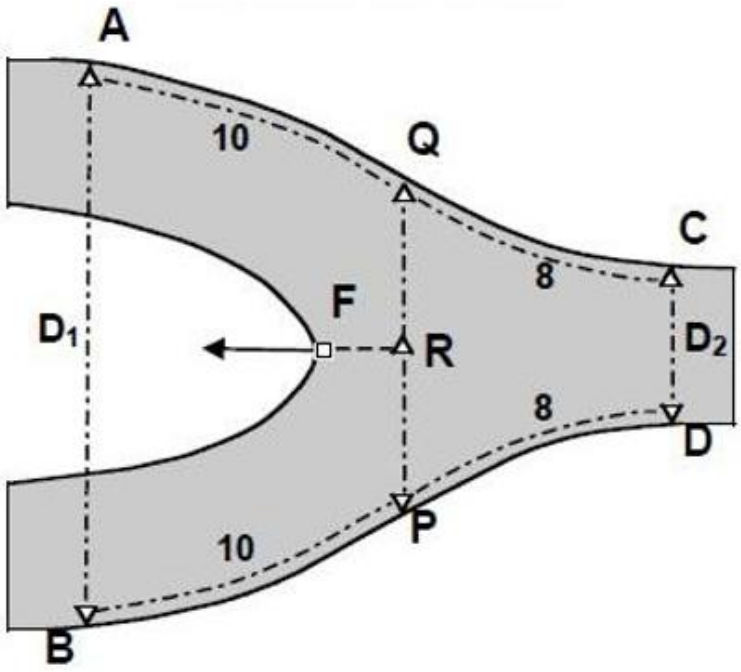


Рис. 2.17 - Видалення занадто вузьких мінімумів

Алгоритм 9. Видалення занадто вузьких мінімумів

```
1: procedure Removal of too narrow minutiae (F )
2: T = 180 - direction(F)
```

```
   3: R = translate(F, 3 pixels, T)
   4: Q = findEdge(R, Up, 12 pixels)
   5: P = findEdge(R, Down, 12 pixels) 6: Pts = traceContour(Q, 10 pixels) 7: A =
Pts(10)
   8: Pts = traceContour(Q, -8 pixels)
   9: C = Pts(8)
   10:Pts = traceContour(P, 10 pixels)
   11:B = Pts(10)
   12:Pts = traceContour(P, -8 pixels)
   13:D = Pts(8)
   14:D1 = distance(A, B)
   15:D2 = distance(C, D)
   16:if D1/D2 <= 2.25 then
   17:remove(F )
   18:end if
   19: end procedure
```

Попередні тести для кандидатів, які є надто широкими. Цей крок тестує точки, які знаходяться на надто вузьких структурах. Рисунок 2.17. ілюструє цей тест. Починаючи з точки, що дорівнює кандидату F, його координати переводяться на 3 пікселя (PORESTRANSR) навпроти напрямку мінімуму. Потім верхній край та нижні краї огинаючої структури розташовуються на (Q, P). З цих двох точок ребро простежується з лівого 10 пікселів (PORESSTEPFWD) та праворуч 8 пікселів (PORESSTEPBWD). Точки в кінці контурів 10 пікселів зберігаються (A, B), а точки в кінці 8 піксельних контурів зберігаються (C, D). Далі обчислюються відстані між цими парами точок, а співвідношення (D1 / D2) обчислюється. Якщо коефіцієнт становить 2,25 (PORESMAXRATIO), тоді пункт "мінімум" видаляється з списку кандидатів. Фактично, якщо процес не може знайти жодного з пунктів на ілюстрації, то кандидат мінусів вилучається.

2.9 Граф сусідських вершин

Допоміжна інформація зазвичай включає напрям мініатюри, його тип, і може містити інформацію, що стосується сусідів, що знаходяться за межами графу. Поза межами позиції, напрямку та типу, не існує стандартної схеми сусідів. Різні системи AFIS використовують різні сусідні топології та атрибути. Одним з загальних атрибутів є кількість пропущених хребтів (названих хребтовими переходами) між мініатюрами та кожним із сусідів. Наприклад,

IAFIS [8] використовує хребет перетинання між мінімумаи і його 8 найближчима сусідами, де кожен сусід є найближчим у вказаному октанті. Отримують до 5 найближчих сусідів (MAXNBRS). Враховуючи точку мінімуму, виділяються найближчі сусіди, розташовані нижче (у тій же піксельній колонці) і справа (у межах всього стовпця пікселя) у зображенні. Ці найближчі сусіди сортуються в порядку їхнього напрямку, починаючи з вертикальної та за годинниковою стрілкою. Використовуючи цю топологію, кількість хребтів обчислюється та фіксується між точкою мінімуму і кожним із найближчих сусідів.

2.10 Оцінка якості отриманих даних

Однією з цілей розробки цього програмного забезпечення була оцінка рівня якості / надійності, яка була пов'язана з кожною виявленою точкою мінімуму. Навіть із тривалим переліком перерахованих вище кроків, помилкові мінуси потенційно залишаються у списку кандидатів. Поліпшений якісний показник може допомогти в управлінні цим процесом, тому що помилкові мініатюри мають бути нижчими за якість, ніж реальні. Завдяки динамічному пороговому значенню можна визначити компроміс між збереженням помилкових мінусів та викидами справжніх мінімумів.

Два чинники поєднують в собі якісний показник для кожної виявленої точки. Перший коефіцієнт - L, береться безпосередньо з місця розташування точки мінімуму на карті якості. Спочатку призначається один з п'яти рівнів якості, 4 - найвища, а найнижча - 0. Другий фактор базується на простій статистиці інтенсивності пікселів (середнє та стандартне відхилення) у безпосередній близькості від точок мінімуму. Розмір сусідства встановлено до 11 пікселів (RADIUSMM). Це досить велике значення, щоб утримувати частини середнього хребта та долини. Висока якість в зображенні з відбитком пальців матиме суттєвий контраст, який покриває весь спектр відтінків сірого. Отже, середня інтенсивність пікселя буде дуже близькою до 127. З подібних причин інтенсивність пікселів ідеального сусідства матиме стандартне відхилення більше або дорівнює 64. Використовуючи цю логіку, наступна міра надійності -

R, обчислюється середнє середнє значення та стандартне відхилення. Значення якості буде знаходится в діапазоні від 0,01 до 0,99. Низька якість означає мінімум, виявлену в нижчих регіонах зображення, тоді як висока якість - це мінімум, виявлений у регіоні вищої якості.

2.11 Вихідні мінімуми

Після завершення, результуючі мінімуми виводяться у файл. Якщо вхідний файл був файлом формату ANSI / NIST, автоматично додається два нових записи та дані записуються новий файл формату ANSI / NIST до oroot.mdt, де oroot передається як параметр для наміру. Нові записи - це запис типу 9, що містить виявлені мініатюри, побудований та вставлений разом із записом типу 13 або типу 14, що зберігає результати бінаризації зображень. Якщо вхідне зображення має прихований відбиток, то результати бінаризації зберігаються в запису типу 13; інакше результати зображення зберігаються в запису типу 14. Слід зазначити, що мініатюри у запису Type-9 форматуються у полях NIST відповідно до стандарту ANSI / NIST.

Якщо вхідний файл не в форматі ANSI / NIST, отримані мініатюрні елементи можна проаналізувати в текстовому файлі oroot.min, а вихідний файл ANSI / NIST не створювати, але створити пустий піксельний файл із результатами бінаризації зображення. Для всіх типів вводу виявлені мініатюри також записуються у текстовий файл oroot.xyt, який відформатовано для використання з matzger bozorth3. Цей файл має одну лінію, розділену пробілом, що містять координати x і y, кут нахилу тета та мінімальну якість. Вихідні мінімуми знаходяться у форматі ANSI / NIST, який має початок у нижній лівій частині зображення та в напрямках, що вказують і віддаляють від кінця хребта або біфуркації. Існує можливість виведення мініатюри у представленні M1 (ANSI INCITS 378-72 3004), яке має піксельний потік у верхньому лівому куті зображення та вказівки до кінця хребта або біфуркаційної долини. Останній вихідний текстовий файл oroot.min, містить форматований список атрибутів, пов'язаних із кожним виявленим мінус-елементом у зображенні відбитків

пальців. Серед цих атрибутів розташовано піксельну координату мінімум, її напрямок та тип.

2.12 Відстеження відбитків пальців

Алгоритм і забезпечення для прийому функцій, витягнутих з двох відбитків пальців (виявлення мікрофільмів) та їх з'єднання разом для цілей особистої перевірки або ідентифікації "один на один". Цей тип алгоритму зазвичай називають матрицею відбитків пальців.

Алгоритм співставлення BOZORTH3 [9] обчислює показник відповідності між хвилинами від будь-яких двох відбитків пальців, щоб визначити, чи є вони з того самого пальця. Це модифікований варіант відстеження відбитків пальців, написаний Аллан С. Боцортом під час роботи в ФБР. Рання версія відповідного алгоритму, яку NIST використала всередині, називалася bozorth98. Механізм BOZORTH3 функціонально аналогічний тому, що використовує bozorth98. Він був вдосконалений видаленням помилок у коді та підвищенням швидкості макета.

Метод BOZORTH3 використовує лише розташування (x, y) та орієнтацію мініатюри, який вказує на відповідність відбитків пальців. Механізм є інваріантним для обертання та перекладу. Матчер створює окремі таблиці для відповідності відбитків пальців, які визначають відстань та орієнтацію між мінімумами на кожному відбитку пальців. Ці дві таблиці потім порівнюються для сумісності та будується нова таблиця, в якій зберігається інформація, що відображає сумісність між відбитками пальців. Таблиця сумісності між пальцями використовується для створення підрахунку відповідностей, переглядаючи розмір та кількість сумісних кластерів мінімумів. Нижче описано докладний опис алгоритму BOZORTH3.

2.12.1 Алгоритм Bozorth

Важливо відзначити два основні моменти стосовно датчика відбитків пальців:

1. Функції Minutia [10] ексклюзивно використовуються та обмежуються

розташуванням (x, y) та орієнтацією t, представленими як x, y, t.

2. Алгоритм призначений для інваріантності обертання та перекладу.

Алгоритм складається з трьох основних кроків:

- Побудова таблиці зіставлення внутрішніх відбитків пальців.
- Побудова таблиці сумісності відбитків пальців.
- Перетин в таблиці сумісності між відбитками пальців.

2.12.2 Побудова таблиці відносних вимірювань

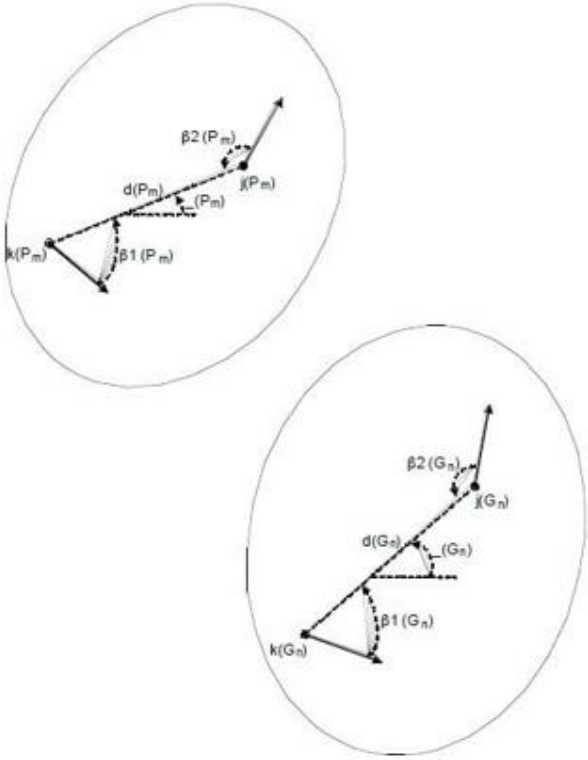


Рис. 2.18 - Виміри

Першим кроком в Bozorth скануванні є обчислення відносних вимірювань від кожного мінімума на відбитку пальців на всі інші мініатюри на одному і тому ж просторі. Ці відносні вимірювання зберігаються в порівняльній таблиці мінімумів і це забезпечує поворот алгоритмів та інваріантність перекладу. На рисунку 2.18. ілюструються вимірювання міжрядкових мікросхем, які використовуються у алгоритмі. У цьому прикладі показано два мінуси. Мінус k знаходиться у нижньому лівому куті відбитка пальця, що зображує розташування (xk, yk), а лінія спрямована вниз і вправо, що відображає

орієнтацію tk. Друга мінус j знаходиться у верхньому правому куті з орієнтацією вгору і вправо. Щоб визначити відносне поступальне положення, відстань dkj обчислюється між двома розташуваннями мінімуму. Ця відстань залишатиметься відносно постійною між відповідними точками на двох різних зображеннях пальця.

Метою для кожного з мінімумів у порівняльному співвідношенні є обчислення кута між кожною мінливістю орієнтації та проміжною лінією. Таким чином, ці кути залишаються відносно постійними до проміжної лінії. У наведеній вище ілюстрації кут kj проміжної лінії між мінімумами k та j обчислюється, беручи арктангенс нахилу проміжної лінії. Кути і j обчислюються відносно проміжної лінії, як показано внаслідок включення kj та кожної орієнтації мінімуму t. Слід зазначити, що точне порівняння проводиться на позиціях мінімуму, відсортованих спочатку по x координаті, потім по y координаті, і що всі орієнтації обмежуються періодом (-4572 мм, 4572 мм). Записи зберігаються у порівняльній таблиці в порядку збільшення відстані, а таблиця обрізається в точці, в якій досягнута максимальна відстань. Здійснюючи ці вимірювання між парами мінусів, порівнювана таблиця повинна бути побудована для кожного з відбитків пальців.

2.12.3 Побудова таблиці зіставлення з відбитками пальців

Наступним кроком в алгоритмі співставлення Bozorth є прийняття таблиць порівняння мінімумів з двох окремих відбитків пальців та пошук сумісних записів між двома таблицями. На рисунку 2.19. зображені два відбитки одного і того ж пальця з невеликими відмінностями: обертання і масштабу. На кожному відбитку відображаються дві відповідні точки. Виміри, обчислені з конкретної пари мінімумів у цьому прикладі, зберігалися як m-й запис у таблиці. Точка P (m запис) позначена Pm. Позначення окремих значень, збережених у таблиці, представлено як функції пошуку за даною таблицею. Наприклад, індекс нижньої лівої мініатюри зберігається у таблиці Pm і називається k (Pm), тоді як відстань між двома мініатюрами також зберігається у таблиці Pm і називаються d (Pm).

Нижній правий відбиток являє собою друк та використовує подібні позначення, за винятком того, що всі його порівняльння за парою були збережені в таблиці G, а розміри, зроблені на двох відповідних мініатюрах у друку галереї, були збережені в таблиці Gn . Виконано наступні три тести, щоб визначити, чи таблиці записів Pm та Gn сумісні. Перший тест перевіряє, чи відповідні відстані перебувають у межах Td. Останні два тести перевіряють, чи мають відносні кутові мінімуми в межах заданого допущення T.

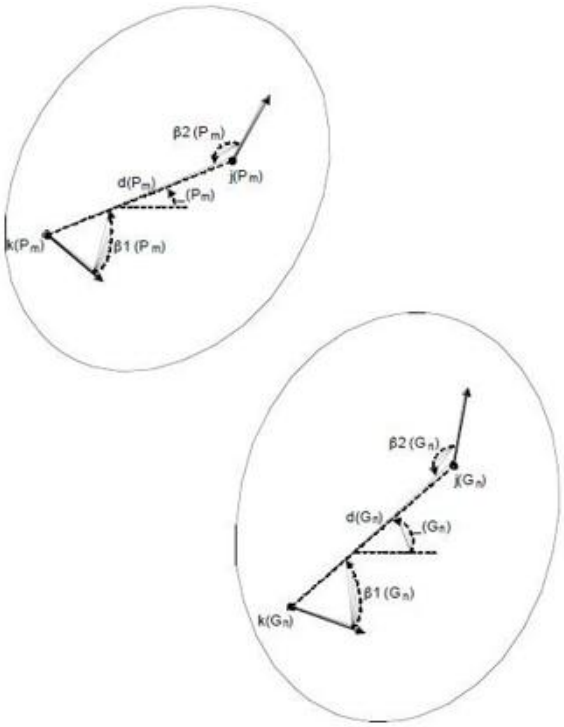


Рис. 2.20 - Вимірювання внутрішніх мінімумів

Якщо відстань відносна та кут мінімуму між двома таблицями перебуває у межах допустимих значень, то в таблицю сумісності додається наступний запис: таким чином, вступ до таблиці сумісності включає дві пари мініатюри, одна пара з датчика відбитка пальця (k (Pm), j (Pm)), а інший від відбитка пальця (k (Gn), j (Gn)). Введення в таблицю сумісності вказує на те, що k (Pm) відповідає k (Gn), а j (Pm) відповідає j (Gn).

2.12.4 Створення таблиці сумісності

На цьому етапі створюється таблиця сумісності, яка складається з переліку

асоціацій сумісності між двома парами потенційно відповідних мініатюр. Ці асоціації являють собою єдині посилання в графіку сумісності. Щоб визначити, наскільки добре оброблені два відбитки пальців, необхідно перейти до графу сумісності, щоб знайти найдовший шлях пов'язаних об'єднань сумісностей. Точка збігу буде асоціюватися з точкою найдовшого шляху. Існує кілька серйозних проблем для такого простого підходу. До них відносяться:

1. Таблиця сумісності не є когерентним графіком, а скоріше непересічним набором окремих посилань у графі.
2. Кожен вузол графа потенційно пов'язаний з багатьма іншими вузлами.
3. На графіку немає очевидного кореневого вузла, який можна передбачити, щоб привести до максимального шляху.
4. Порожнечі в межах одного з двох відбитків пальців, що підлягають узгодженню, призведуть до розриву на графіку.

Щоб обійти ці проблеми було створення рішення, яке використовуються в данній роботі. Коли проводяться переходи, кластери графіку сумісності створюються шляхом об'єднання записів у таблиці. Після завершення переходів комбіновані кластери об'єднуються і кількість пов'язаних табличних записів по комбінованих кластерах накопичується, щоб сформувати відповідності. Чим більше число пов'язаних об'єднань сумісності, тим більший показник відповідності.

2.13 Продуктивність

Результат в таблиці 2.1. показує загальний час, необхідний для перетворення цілої бази даних (NIST DB2, DB3, DB4) зображень з одного формату в інший та загальний час для індексування усієї бази даних.

Таблиця 2.1 - Повний тест швидкості індексування DB4, DB3, DB2

	NITRklOld:	NITRklNew:	DB4:	DB3:	DB2:
On	11:38:29.136	11:39:17.313	10:15:27.373	10:18:32.542	10:22:02.218
To24	11:38:53.524	11:39:37.204	10:17:03.014	10:20:24.178	10:23:24.044
ToWSQ	11:39:17.309	11:39:58.169	10:18:32.539	10:22:02.212	10:25:45.041
Resol. Items Format	288x320 132x2 BMP	288x320 130x2 BMP	288x384 100x8 TIF	300x480 100x8 TIF	328x364 100x8 TIF
Net Index Time	On: 04:54:53.685 Off: 04:56:11.065	On: 04:53:18.615 Off: 04:54:53.673	On: 04:16:56.845 Off: 04:21:13.979	On: 04:06:53.727 Off: 04:14:18.124	On: 04:14:18.134 Off: 04:16:56.647

Наступний результат на рисунку 2.21 показує споживання процесора та споживання пам'яті під час перетворення зображення та індексування бази даних.

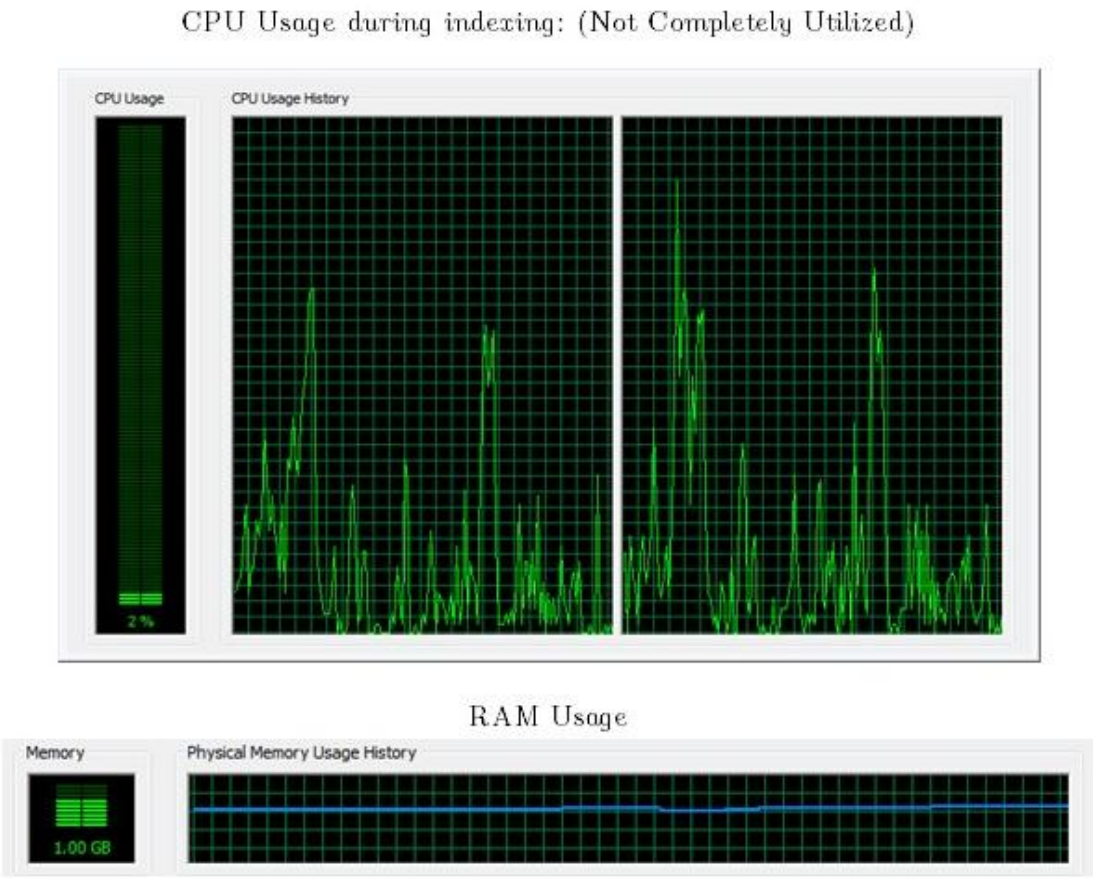


Рис. 2.21 - Споживання процесора

Наступний результат на рисунку 2.22 час від початку до закінчення підбору 1 відбитка пальця у групі з 72 відбитків пальців. Перевірено всі 44 зразки.

Sample 1:72	Start	Finish	Delta	Match Found?
A001.AN2.xyt	18:17:10.512	18:17:10.612	0:00:00.100	Y
A002.AN2.xyt	18:17:11.720	18:17:11.722	0:00:00.002	Y
A003.AN2.xyt	18:17:12.618	18:17:12.620	0:00:00.002	Y
A004.AN2.xyt	18:17:13.317	18:17:13.319	0:00:00.002	Y
A005.AN2.xyt	18:17:14.089	18:17:14.091	0:00:00.002	Y
A006.AN2.xyt	18:17:14.717	18:17:14.719	0:00:00.002	Y
A007.AN2.xyt	18:17:15.568	18:17:15.570	0:00:00.002	Y
A008.AN2.xyt	18:17:16.898	18:17:16.900	0:00:00.002	Y
A009.AN2.xyt	18:17:17.631	18:17:17.633	0:00:00.002	Y
A010.AN2.xyt	18:17:18.932	18:17:18.934	0:00:00.002	Y
A011.AN2.xyt	18:17:19.612	18:17:19.614	0:00:00.002	Y
A012.AN2.xyt	18:17:20.346	18:17:20.347	0:00:00.001	Y
A013.AN2.xyt	18:17:21.424	18:17:21.426	0:00:00.002	Y
A014.AN2.xyt	18:17:22.351	18:17:22.353	0:00:00.002	Y
A015.AN2.xyt	18:17:23.405	18:17:23.407	0:00:00.002	Y
A016.AN2.xyt	18:17:24.442	18:17:24.443	0:00:00.001	Y
A017.AN2.xyt	18:17:25.343	18:17:25.345	0:00:00.002	Y
A018.AN2.xyt	18:17:26.346	18:17:26.348	0:00:00.002	Y
A019.AN2.xyt	18:17:27.485	18:17:27.487	0:00:00.002	Y
A020.AN2.xyt	18:17:28.332	18:17:28.334	0:00:00.002	Y
A021.AN2.xyt	18:17:29.293	18:17:29.295	0:00:00.002	Y
A022.AN2.xyt	18:17:30.024	18:17:30.026	0:00:00.002	Y
A023.AN2.xyt	18:17:30.733	18:17:30.735	0:00:00.002	Y
A024.AN2.xyt	18:17:31.581	18:17:31.583	0:00:00.002	Y
A025.AN2.xyt	18:17:32.851	18:17:32.854	0:00:00.003	Y
A026.AN2.xyt	18:17:34.241	18:17:34.243	0:00:00.002	Y
A027.AN2.xyt	18:17:35.055	18:17:35.057	0:00:00.002	Y
A028.AN2.xyt	18:17:36.963	18:17:36.964	0:00:00.001	Y
A029.AN2.xyt	18:17:37.808	18:17:37.810	0:00:00.002	Y
A030.AN2.xyt	18:17:39.490	18:17:39.492	0:00:00.002	Y
A031.AN2.xyt	18:17:40.232	18:17:40.234	0:00:00.002	Y
A032.AN2.xyt	18:17:41.239	18:17:41.241	0:00:00.002	Y
A033.AN2.xyt	18:17:41.932	18:17:41.934	0:00:00.002	Y
A034.AN2.xyt	18:17:42.977	18:17:42.979	0:00:00.002	Y
A035.AN2.xyt	18:17:44.170	18:17:44.172	0:00:00.002	Y
A036.AN2.xyt	18:17:45.552	18:17:45.554	0:00:00.002	Y
A037.AN2.xyt	18:17:46.723	18:17:46.725	0:00:00.002	Y
A038.AN2.xyt	18:17:47.582	18:17:47.584	0:00:00.002	Y
A039.AN2.xyt	18:17:47.995	18:17:47.997	0:00:00.002	Y
A040.AN2.xyt	18:17:48.803	18:17:48.805	0:00:00.002	Y
A041.AN2.xyt	18:17:49.295	18:17:49.297	0:00:00.002	Y
A042.AN2.xyt	18:17:49.659	18:17:49.661	0:00:00.002	Y
A043.AN2.xyt	18:17:49.938	18:17:49.940	0:00:00.002	Y
A044.AN2.xyt	18:17:50.573	18:17:50.575	0:00:00.002	Y

Рис. 2.22 - Тестування відбитка пальця на DB2

Наступний результат на рисунку 2.23 показує бал відповідності 1 відбитку в групі з 72 відбитків пальців.

```
10 A001.AN2.xyt A002.AN2.xyt
581 A002.AN2.xyt A002.AN2.xyt
5 A003.AN2.xyt A002.AN2.xyt
5 A004.AN2.xyt A002.AN2.xyt
5 A005.AN2.xyt A002.AN2.xyt
7 A006.AN2.xyt A002.AN2.xyt
7 A007.AN2.xyt A002.AN2.xyt
9 A008.AN2.xyt A002.AN2.xyt
8 A009.AN2.xyt A002.AN2.xyt
5 A010.AN2.xyt A002.AN2.xyt
6 A011.AN2.xyt A002.AN2.xyt
8 A012.AN2.xyt A002.AN2.xyt
6 A013.AN2.xyt A002.AN2.xyt
10 A014.AN2.xyt A002.AN2.xyt
9 A015.AN2.xyt A002.AN2.xyt
7 A016.AN2.xyt A002.AN2.xyt
7 A017.AN2.xyt A002.AN2.xyt
9 A018.AN2.xyt A002.AN2.xyt
9 A019.AN2.xyt A002.AN2.xyt
8 A020.AN2.xyt A002.AN2.xyt
6 A021.AN2.xyt A002.AN2.xyt
7 A022.AN2.xyt A002.AN2.xyt
9 A023.AN2.xyt A002.AN2.xyt
11 A024.AN2.xyt A002.AN2.xyt
11 A025.AN2.xyt A002.AN2.xyt
6 A026.AN2.xyt A002.AN2.xyt
13 A027.AN2.xyt A002.AN2.xyt
6 A028.AN2.xyt A002.AN2.xyt
14 A029.AN2.xyt A002.AN2.xyt
6 A030.AN2.xyt A002.AN2.xyt
6 A031.AN2.xyt A002.AN2.xyt
8 A032.AN2.xyt A002.AN2.xyt
10 A033.AN2.xyt A002.AN2.xyt
8 A034.AN2.xyt A002.AN2.xyt
```

Рис. 2.23 - Результат тестування відбитків пальців з балами на DB2

Наступний результат на рисунку 2.24. показує бал відповідності 1 відбитку в групі 72 відбитків пальців.

```
6 A001.AN2.xyt A003.AN2.xyt
5 A002.AN2.xyt A003.AN2.xyt
505 A003.AN2.xyt A003.AN2.xyt
8 A004.AN2.xyt A003.AN2.xyt
6 A005.AN2.xyt A003.AN2.xyt
8 A006.AN2.xyt A003.AN2.xyt
6 A007.AN2.xyt A003.AN2.xyt
5 A008.AN2.xyt A003.AN2.xyt
6 A009.AN2.xyt A003.AN2.xyt
6 A010.AN2.xyt A003.AN2.xyt
6 A011.AN2.xyt A003.AN2.xyt
6 A012.AN2.xyt A003.AN2.xyt
9 A013.AN2.xyt A003.AN2.xyt
8 A014.AN2.xyt A003.AN2.xyt
10 A015.AN2.xyt A003.AN2.xyt
5 A016.AN2.xyt A003.AN2.xyt
6 A017.AN2.xyt A003.AN2.xyt
6 A018.AN2.xyt A003.AN2.xyt
5 A019.AN2.xyt A003.AN2.xyt
10 A020.AN2.xyt A003.AN2.xyt
6 A021.AN2.xyt A003.AN2.xyt
10 A022.AN2.xyt A003.AN2.xyt
12 A023.AN2.xyt A003.AN2.xyt
7 A024.AN2.xyt A003.AN2.xyt
7 A025.AN2.xyt A003.AN2.xyt
7 A026.AN2.xyt A003.AN2.xyt
7 A027.AN2.xyt A003.AN2.xyt
6 A028.AN2.xyt A003.AN2.xyt
9 A029.AN2.xyt A003.AN2.xyt
5 A030.AN2.xyt A003.AN2.xyt
8 A031.AN2.xyt A003.AN2.xyt
8 A032.AN2.xyt A003.AN2.xyt
5 A033.AN2.xyt A003.AN2.xyt
12 A034.AN2.xyt A003.AN2.xyt
```

Рис. 2.24 Повторний результат тестування відбитків пальців з балами на DB2

2.13.1 Експеримент у реальному часі

Короткий зміст модуля реєстрації відбитків пальців показано на рисунку 2.25

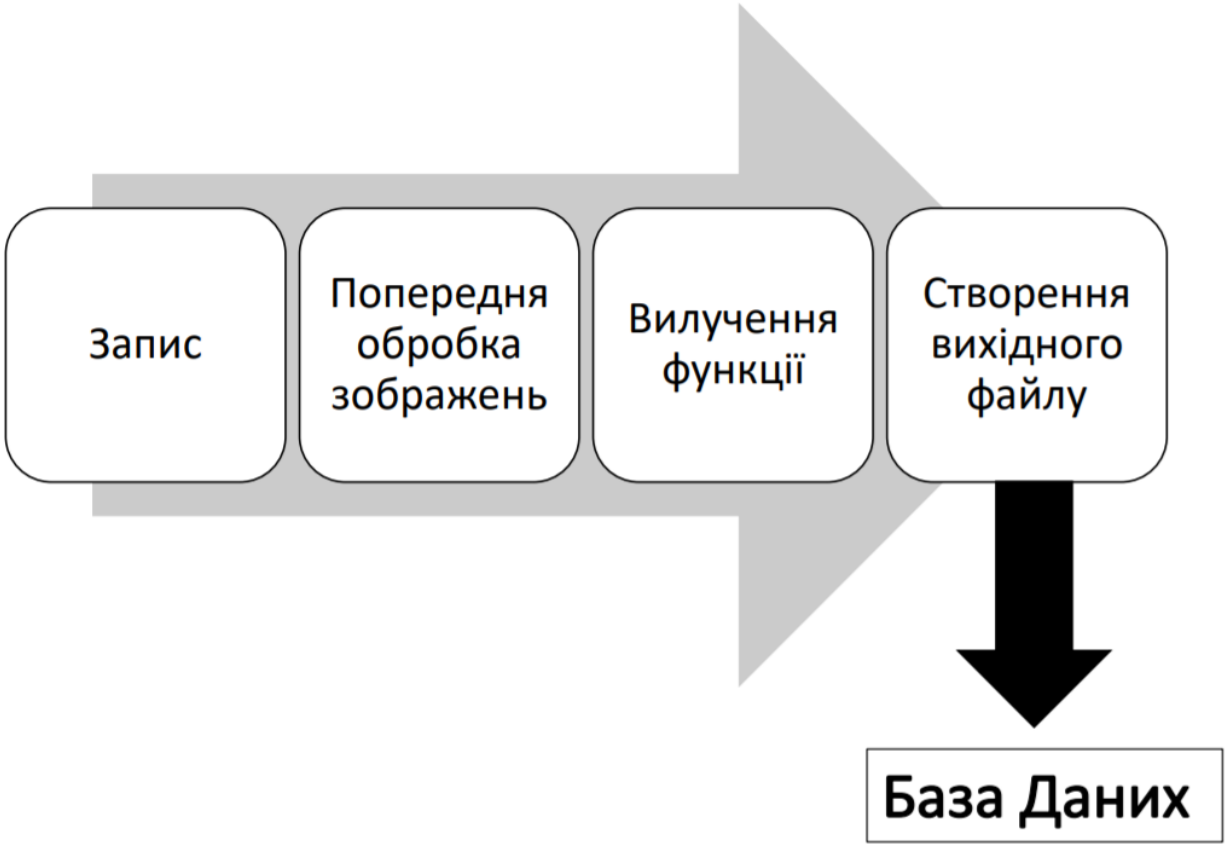


Рис. 2.25 - Схема реєстрації відбитків пальців

У реальному часі першим кроком є захоплення зображення, у роботі використано зчитувач, щоб сфотографувати зображення відбитків пальців. Приклад зчитувача зображено на рисунку 2.26.

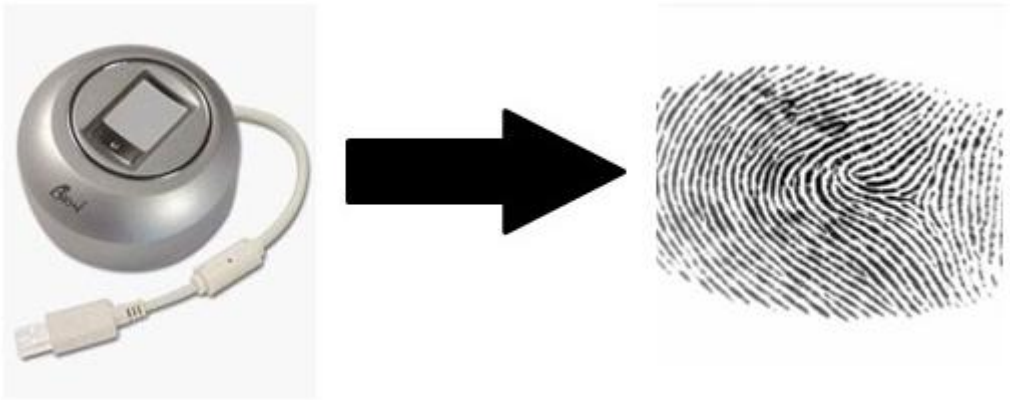


Рис. 2.26 - Зчитувач відбитків пальців

На рисунку 2.27. можна побачити візуальний висновок процедур попередньої обробки.

Рис. 2.27 - Попередня обробка вхідного зображення

Візуальний вивід, який генерується при моделюванні модуля виявлення мінімумів. На рисунку 2.28 можна побачити візуальний висновок про те, як витягуються мінімуми.

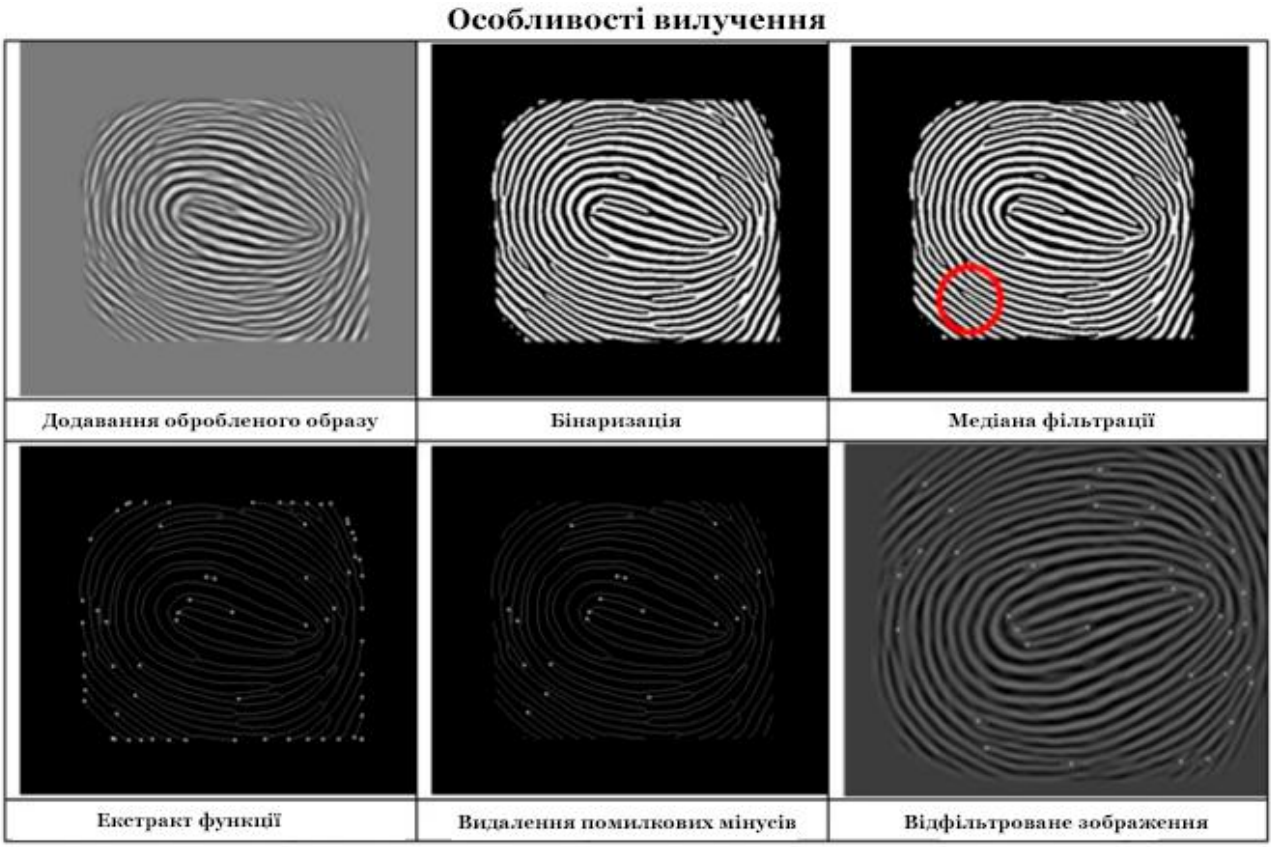


Рис. 2.28 - Процес вилучення функції і створення відфільтрованого зображення

Рисунок 2.29. описує, який вихідний файл буде зберігатися з вибраних мініатюр.

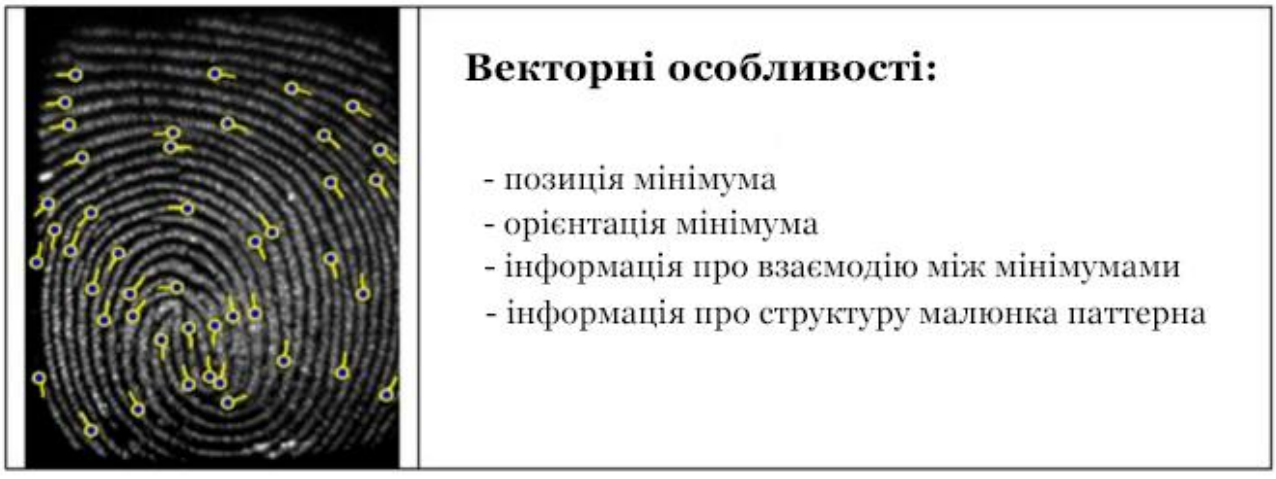


Рис. 2.29 - Збереження вектора об'єктів у вихідному файлі

Вихідні дані відбитків пальців зберігаються у спеціально визначеному векторному файлі vikxyt, який закодований і стиснутий з міркувань безпек. Збір таких файлів показаний на рисунку 2.30.

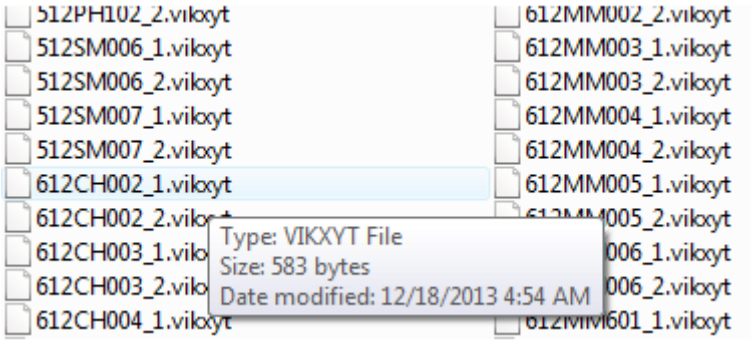


Рис. 2.30 - Вектори, що зберігаються у файлі

3. РЕАЛІЗАЦІЯ МЕТОДІВ ІДЕНТИФІКАЦІЇ НА BLOCKCHAIN ПРОЕКТАХ

3.1 Технологія Blockchain

Блок транзакцій — спеціальна структура для запису групи транзакцій в системі Біткоїн та аналогічних їй[10].

Щоб транзакція вважалася достовірною («підтвердженою»), її формат і підписи повинні перевірити і потім групу транзакцій записати в спеціальну структуру — блок. Інформацію в блоках можна швидко перевірити. Кожен блок завжди містить інформацію про попередній блок. Усі блоки можна вибудувати в один ланцюжок, який містить інформацію про всі вчинені коли-небудь операції з біткоїнами. Перший блок в ланцюжку — первинний блок (англ. genesis block) — розглядається як окремий випадок, оскільки в нього відсутній материнський блок[11].

Блок складається із заголовка та списку транзакцій. Заголовок блоку включає в себе свій хеш, хеш попереднього блоку, хеши транзакцій та додаткову службову інформацію. Першою транзакцією в блоці завжди вказується отримання комісії, яка стане нагородою користувачеві за створений блок.

Далі йдуть всі або деякі з останніх транзакцій, які ще не були записані в попередні блоки. Для транзакцій в блоці використовується деревоподібне хешування, аналогічне формуванню хеш-суми файлу в протоколі BitTorrent. Транзакції, крім нарахування комісії за створення блоку, містять всередині атрибута input посилання на транзакцію, за якою на цей рахунок були отримані біткоїни. Комісійні операції можуть містити в атрибуті будь-яку інформацію (для них це поле носить назву англ. Coinbase parameter), оскільки у них немає батьківських транзакцій.

Створений блок буде прийнятий іншими користувачами, якщо числове значення хешу заголовка менше або дорівнює певному числу, величина якого періодично коригується. Оскільки результат хешування (функції SHA-256) необоротний, немає алгоритмуотримання бажаного результату, окрім

випадкового перебору. Якщо хеш не задовольняє умову, то довільно змінюється блок службової інформації в заголовку, і хеш перераховується. Зазвичай потрібна велика кількість перерахунків. Коли варіант знайдено, вузол розсилає отриманий блок іншим підключеним вузлам, які перевіряють блок. Якщо помилок немає, то блок вважається доданим в ланцюжок, і наступний блок повинен включити в себе його хеш.

Величина цільового числа, з яким порівнюється хеш, коригується через кожні 2016 блоків. Заплановано, що вся мережа буде витрачати на генерацію одного блоку приблизно 10 хвилин, на 2016 блоків — близько двох тижнів. Якщо 2016 блоків сформовані швидше, то мета трохи зменшується і досягти її стає важче, в іншому випадку мета збільшується. Зміна складності обчислень не впливає на надійність мережі Біткоїн і потрібна лише для того, щоб система генерувала блоки майже з постійною швидкістю, що не залежить від потужності мережі.

3.1.1 Ланцюжок блоків

Блоки одночасно формуються безліччю «майнерів». Блоки, які задовольняють критерії, відправляються в мережу, включаючись у розподілену базу блоків. Регулярно виникають ситуації, коли кілька нових блоків в різних частинах розподіленої мережі називають попереднім один і той же блок, тобто ланцюжок блоків може розгалужуватися. Спеціально чи випадково можна обмежити ретрансляцію інформації про нові блоки (наприклад, один з ланцюжків може розвиватися в рамках локальної мережі). У цьому випадку можливе паралельне нарощування різних гілок.

У кожному з нових блоків можуть траплятися як однакові транзакції, так і різні, що входять тільки в один з них. Коли ретрансляція блоків поновлюється, майнери починають вважати головним ланцюжок з урахуванням рівня складності хешу і довжини ланцюжка. При рівності складності і довжини перевага віддається тому ланцюжку, кінцевий блок якого з'явився раніше.

Транзакції, що увійшли тільки у відхилену гілку (в тому числі з виплати винагороди), втрачають статус підтверджених.

Якщо це операції з передачі біткоїнів, то вона буде поставлена в чергу і потім включена в черговий блок. Транзакції отримання винагороди за створення відсічених блоків не дублюються в іншій гілці, тобто «зайві» біткоїни, виплачені за формування відсічених блоків, не отримують подальших підтверджень і «втрачаються»[12].

Розподілена база даних Blockchain формується як безперервно зростаючий ланцюжок блоків з записами про всі транзакції. Копія бази даних або її частини одночасно зберігаються на безлічі комп'ютерів та синхронізуються відповідно до формальних правил побудови ланцюжка блоків. Інформація в блоках не шифрована і доступна у відкритому вигляді, однак захищена від змін криптографічно через хеш-ланцюжок.

Найчастіше умисна зміна інформації в будь-якій з копій бази або навіть в досить великій кількості копій не буде визнана істинною, оскільки не відповідатиме правилам. Деякі зміни можуть бути прийняті, якщо будуть внесені в усі копії бази (наприклад, видалення кількох останніх блоків через помилку в їхньому формуванні).

3.1.2 Підтвердження транзакцій

Поки транзакція не включена в блок, система вважає, що кількість біткоїнів за якоюсь адресою залишається незмінною. У цей час є технічна можливість оформити кілька різних транзакцій для передачі з однієї адреси одних і тих же біткоїнів різним одержувачам[13]. Але як тільки одна з подібних транзакцій буде включена в блок, то інші транзакції з цими ж біткоїнами система вже буде ігнорувати.

Наприклад, якщо в блок буде включена більш пізня транзакція, то більш рання буде вважатися помилковою. Є невелика ймовірність, що при розгалуженні дві подібні транзакції потраплять в блоки різних гілок. Кожна з них буде вважатися правильною, лише при відмиранні гілки одна з транзакцій стане

вважатися помилковою. При цьому не буде мати значення час здійснення операції.

Отож попадання транзакції в блок є підтвердженням її достовірності незалежно від наявності інших транзакцій з тими ж біткоїнами. Кожен новий блок вважається додатковим підтвердженням транзакцій з попередніх блоків. Якщо в ланцюжку три блоки, то транзакції з останнього блоку будуть підтверджені один раз, а поміщені в перший блок будуть мати три підтвердження. Досить дочекатися декількох підтверджень, щоб звести ймовірність скасування транзакції до мінімуму.

Для зменшення впливу таких ситуацій на мережу існують обмеження на розпорядження щойно отриманими біткоїнами. Згідно сервісу blockchain.info до травня 2015 року максимальна довжина відкинутих ланцюжків була 5 блоків. Необхідне число підтверджень для розблокування отриманого залежить від програми-клієнта або від вказівок приймаючої сторони. Клієнт «Bitcoin-qt» для відправлення не потребує наявності підтверджень, однак у більшості одержувачів за замовчуванням виставлено вимогу 6 підтверджень, тобто реально скористатися отриманим зазвичай можна через годину. Різні онлайн-сервіси часто встановлюють свій поріг підтверджень.

Біткоїни, отримані за створення блоку, протокол дозволяє використовувати після 100 підтверджень[14], але стандартна програма-клієнт показує комісію через 120 підтверджень, тобто зазвичай скористатися комісією можна приблизно через 20 годин після її нарахування.

3.2 Складність

Оскільки обчислювальні потужності мережі непостійні, цей параметр перераховується клієнтами мережі через кожні 2016 блоків таким чином, щоб підтримувати середню швидкість формування розподіленої БД на рівні 2016 блоків в два тижні. Таким чином 1 блок повинен створюватися приблизно раз на десять хвилин. На практиці, коли обчислювальна потужність мережі зростає — відповідні часові проміжки коротше, а коли знижується — довший[15].

Перерахунок складності з прив'язкою до часу можливий завдяки наявності в заголовках блоків часу їх створення. Воно записано в Unix-форматі і взято за системним годинником автора блоку (якщо блок створений у пулі, то за системним годинником сервера цього пулу)

3.3 Архітектура клієнт-сервер

Архітектура клієнт-сервер - архітектура комп'ютерної мережі, в якій безліч клієнтів (віддалених процесорів) запитують і отримують доступ до послуг від сервера. Клієнт і сервер взаємодіють між собою за допомогою різних мережевих протоколів, таких як IP протокол, HTTP протокол, FTP та інші. Вибір протоколу взаємодії залежить від сфери його застосування. Вибір протоколу взаємодії залежить від сфери його застосування. Комп'ютери клієнтської сторони надають інтерфейс, що дозволяє користувачеві комп'ютера запитувати доступ до послуг сервера і виводити результати, отримані від сервера. Сервер очікує, поки надійдуть запити від клієнтів, і потім відповідає на них. В ідеалі, сервер надає стандартизований зрозумілий інтерфейс клієнтам, щоб клієнт не був обізнаний про специфіку системи (тобто апаратного і програмного забезпечення), яка надає послуги. Клієнтська частина програми найчастіше розташовується на робочих станціях або персональних комп'ютерах, в той час як сервери розташовуються в інших частинах мережі, найчастіше - на більш потужних машинах.

Ця обчислювальна модель особливо ефективна в тому випадку, якщо у клієнтів і сервера є чітко визначені завдання, які вони зазвичай виконують. Безліч клієнтів може звертатися до даних на сервері одночасно, і в той же час комп'ютери клієнтів можуть виконувати інші дії. Так як серверний і клієнтський комп'ютер вважаються інтелектуальними пристроями, модель клієнт-сервер повністю відрізняється від старої мейнфрейм моделі, в якій центральний мейнфрейм-комп'ютер виконував всі завдання для пов'язаних з ним "німих" терміналів.

Клієнтська частина системи, побудованої в рамках даної парадигми, запитує послуги, а сервер - їх представляє. Під послугами в даному випадку

розуміються різні ресурси: дані, файли, об'єкти, час ЦПУ, пристрої відображення і т.д. Для такої системи типові такі властивості:

- Запит на послуги являє собою команду, що потрібно виконати від сервера, і найчастіше абстрактний; сервер сам визначає, якими засобами його виконати;
- Ідеальне клієнт-серверне програмне забезпечення незалежно від операційної системи або апаратної платформи;
- Розташування клієнтів і серверів найчастіше прозоро для користувача;
- Сервер може стати клієнтом, а клієнт - сервером;
- Клієнт-серверне взаємодія починає клієнт, сервер лише відповідає на запити клієнта.

Концепція клієнт-сервер спрямована, перш за все, на поділ навантаження між учасниками процесу взаємодії, а також на розмежування коду замовника послуг (тобто клієнта) і постачальника (тобто сервера). Прийнято виділяти два види архітектури взаємодії клієнт-сервер: двухзвенная архітектура і багаторівнева, яку іноді ще називають трирівневої, проте це окремий випадок.

В рамках двувузлової архітектури три базові компоненти додатка розподіляються між вузлами (клієнтом і сервером) різними способами.

До переваг клієнт-серверної архітектури відносять поділ коду клієнтської і серверної частин, знижені вимоги до комп'ютерів клієнтів (за рахунок виконання більшості обчислювальних операцій на сервері), гнучкість.

Як недоліки даної архітектури слід зазначити високу вартість серверного обладнання (значно перевищує вартість клієнтського), необхідність обслуговування сервера спеціальним співробітником, при недостатній потужності серверного обладнання робота системи дуже помітно погіршується: збільшується час очікування відповіді від сервера, потужності сервера не вистачає для задоволення запитів всіх клієнтів.

Вибір даної архітектури для розроблюваних додатків є досить природним і по суті єдино можливим, виходячи з ідей і вимог. До того ж, дана архітектура дозволяє розширювати і виробляти модифікації функціональної частини програми практично непомітно для кінцевих користувачів, що в процесі експлуатації буде дуже актуально, а в разі використання тонкого клієнта (тобто роботи з додатком через браузер) - повністю приховати внутрішні зміни від користувачів.

3.4 Побудова архітектури системи

На етапі побудови архітектури системи на основі визначених раніше вимог і з урахуванням обраних технологій були розроблені діаграми варіантів використання, компонентів, діяльності та послідовності. Дані діаграми в різних аспектах ілюструють склад і поведінку розроблюваних систем. У цьому розділі розглядається лише кілька прикладів різних видів діаграм, повний перелік діаграм знаходиться в додатку.

ПК Клієнта повинен для доступу до інтернет-сторінці додатка мати встановлений браузер. Також на ПК Клієнта міститься копія Blockchain, яка автоматично скачується при реєстрації і оновлюється при вході в систему.

Сторінка додатки містить керуючі елементи, що дозволяють взаємодіяти з системою і користуватися її функціоналом, відправляючи команди на сервер.

Сервер додатка виконує функції відповідно до отриманих запитами. Функціонал програми умовно розбитий на 4 частини:

- Керування обліковими записами (створення аккаунта користувача і правовласника, автентифікація користувачів і правовласників у системі);
- Управління музикою (реєстрація записів (доступна тільки для правовласників), автентифікація записів);
- Управління метаданими (прикріплення і читання метаданих);
- Управління покупками (здійснення транзакцій).

База даних ClientDB містить дані про зареєстрованих користувачів (таблиця Users) і правовласників (таблиця Rights holders), доступ до бази даних здійснюється за допомогою SQL-запитів з коду серверної частини системи. Таблиця даних про користувачів містить наступні поля:

- Ім'я користувача;
- Хеш-значення пароля;
- Ім'я;
- Прізвище;
- Дата народження.

3.4.1 Діаграми послідовності

Діаграми послідовності призначаються для моделювання взаємодії об'єктів системи [в часі] і обміну повідомленнями між об'єктами. Дані діаграми показують не тільки алгоритм дій, а й унаочнюють «спілкування» елементом системи між собою. Як приклад розглядається діаграма, що ілюструє процес реєстрації користувача в системі.

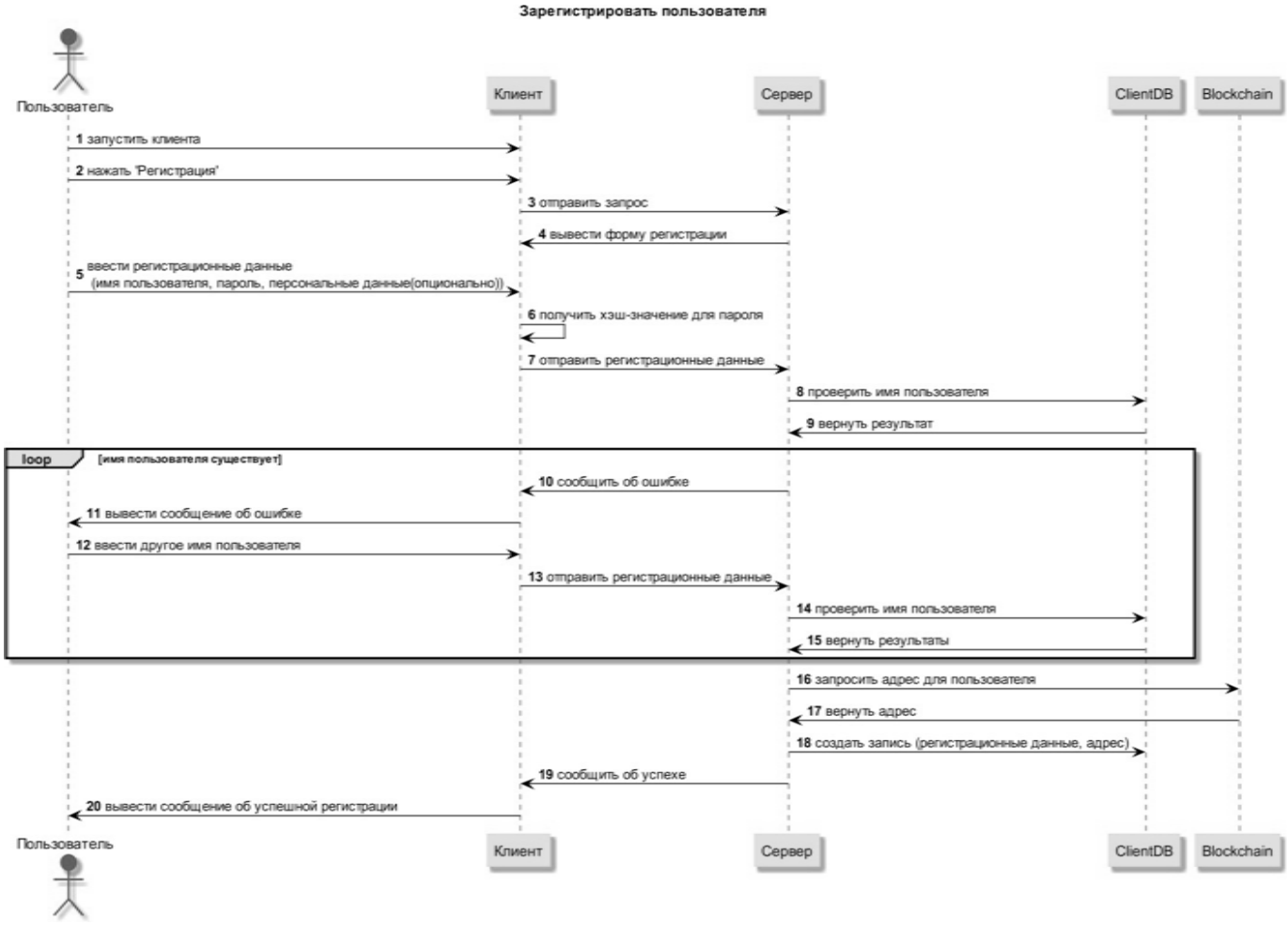

Зарегистрировать пользователя
Пользователь
Клиент
Сервер
ClientDB
Blockchain
1 запустить клиента
2 нажать 'Регистрация'
3 отправить запрос
4 вывести форму регистрации
5 ввести регистрационные данные
(имя пользователя, пароль, персональные данные(опционально))
6 получить хэш-значение для пароля
7 отправить регистрационные данные
8 проверить имя пользователя
9 вернуть результат
loop
[имя пользователя существует]
10 сообщить об ошибке
11 вывести сообщение об ошибке
12 ввести другое имя пользователя
13 отправить регистрационные данные
14 проверить имя пользователя
15 вернуть результаты
16 запросить адрес для пользователя
17 вернуть адрес
18 создать запись (регистрационные данные, адрес)
19 сообщить об успехе
20 вывести сообщение об успешной регистрации
Пользователь
Клиент
Сервер
ClientDB
Blockchain

Рис. 3.1 - Реєстрація користувача (діаграма послідовності)

Користувач запускає клієнта, натискає на кнопку реєстрації. Клієнт відправляє запит реєстрації на сервер, виводиться форма реєстрації. Користувач заповнює поля форми (ім'я користувача, пароль, особисті дані (опціонально)). Генерується хеш-значення для безпечної передачі і зберігання пароля. Реєстраційні дані передаються на сервер. Далі надсилається запит до бази даних на перевірку імені користувача. Якщо користувач з таким ім'ям вже зареєстрований в системі, результат пошуку буде не порожнім, сервер відправить клієнту повідомлення про недоступність імені користувача, клієнт виведе повідомлення для користувача, після чого користувач зможе ввести інше ім'я і процес буде виконаний заново. Далі для користувача генерується blockchain- адреса, створюється запис в базі даних користувачів. Сервер повідомляє клієнту про успішну реєстрацію користувача. Клієнт виводить повідомлення для користувача.

3.4.2 Розробка прототипу функціональної частини системи

В рамках даного етапу роботи були розроблені прототипи частини функціоналу системи для сценарію «Музика і метадані». Розробка здійснювалася на мові програмування Python з використанням технології Multichain, бази даних SQLite, бібліотеки для шифрування hashlib, «оболонки» для виклику JSON-RPC команд з коду програми Savoir. Для тестування клієнт-серверної взаємодії була написана тестова клієнтська і серверна частина системи.

Для перевірки роботи функцій була створена приватна Blockchain, тестові бази даних для зберігання інформації про клієнтів ('clients.db') і записах ('music.db').

Для доступу до Blockchain необхідно в серверній частині системи створити інтерфейс за допомогою бібліотеки Savoir.

```
# Multichain-дані з multichain.conf rpcuser = 'multichainrpc'

rpcpasswd = 'A9gSx6vxLuqHM84axcX5mwq69MB3e3h7iQrSdQ9ECQUd'

rpchost = '10 .129.8.165 '

rpcport = '7220' chainname = 'tchain'

# Створення інтерфейсу для API-команд
```

Після цього можна використовувати створений інтерфейс для того, щоб відправляти команди Blockchain.

Щоб забезпечити безпеку передачі пароля, а також для реєстрації і автентифікації користувачів, правовласників і записів, використовується хешування.

```
# Створення підключення до бази даних

conn_client = sqlite3.connect ('client.db')

# Створення курсора для підключення

cur_client = conn_client.cursor ()
```

Для встановлення зв'язку між клієнтом і сервером були використані сокети.

Приклад створення сокета клієнта:

```
  # Дані з'єднання

  HOST = 'localhost'

  PORT = 9090

  # Створення сокета клієнта

  sock = socket.socket ()

  # Підключення до сервера

  sock.connect ((HOST, PORT))
```

Пример создания сокета сервера:

```
  # Дані з'єднання HOST = ''

  PORT = 9090

  # Створення сокета сервера

  sock = socket.socket ()

  # Зв'язування сокета з хостом і портом

  sock.bind ((HOST, PORT)

  # Визначення обсягу черги вхідних підключень

  sock.listen (10)

  while True:
```

```
# Обробляти вхідні підключення

conn, addr = sock.accept ()

print ('Connected by', addr) while True:

# Поки клієнт не закрив з'єднання # отримувати дані від клієнта

data = sock.recv (1024)
```

Взаємодія між клієнтом і сервером відбувається шляхом відправки команд і параметрів від клієнта серверу і відповідей сервера клієнту. Формування запиту від клієнта і отримання відповіді від сервера відбувається наступним чином:

```
# Формування словника з назвою викликається методу і переданими параметрами і переклад його в рядок
str_json = json.dumps ({'method': <назва методу "," ім'я параметра>: <значення параметра>})
# Новий рядок в байти
data = str_json.encode ()
# Відправка даних на сервер sock.send (data)
# Отримання відповіді від сервера
data = sock.recv (1024)
# Переклад отриманих байт в рядок
data = data.decode ()
```

Обработка запроса клиента и формирование ответа клиенту происходит следующим образом:

```
# Отримання даних від клієнта
data = sock.recv (1024)
# Переклад отриманих байт в рядок
data = data.decode ()
# Новий рядок в словник
dict_json = json.loads (data)
   # Отримати назву методу і параметри зі словника і викликати відповідну функцію
   # Формування відповіді користувачеві в залежності від результатів виконання функції
# Відправка відповіді користувачу
conn.send (answer)
```

У прототипі були реалізовані функції реєстрації користувача і правовласника, автентифікації користувача і правовласника, реєстрації і автентифікації записів. Як приклад написання коду системи буде розглянуто код функції реєстрації користувача для клієнтської і серверної частин системи.

Приклад реалізації функції реєстрації користувача:

```
# На стороні клієнта
username = str (input ( 'Enter username:'))
# Введення імені користувача
```

```
   # Формування і відправка запиту на перевірку доступності імені користувача на
сервер
   # Отримання відповіді сервера
   # Обробка відповіді сервера
   password = string (input ( 'Enter password:'))
   # Введення пароля
   pass_hash = sha256.update (password)
   # Хешування пароля while True:
   # Запит на введення персональних даних
   answer = input ( 'Do you want to enter your personal data? [y / n] \ n'
   # Відмову від введення персональних даних
   first_name = None
   last_name = None birthday = None break
   # Формування і відправка запиту на сервер
   # Отримання відповіді від сервера
   if data == 'Fail':
   # Якщо реєстрація не виконана return False
   else:
   # Якщо реєстрація виконана print (data)
   # Print blockchain address return True
   # На стороні сервера
   # Перед реєстрацією здійснюється перевірка імені користувача
   # Отримання даних від клієнта
   # Створення з'єднання з базою даних і курсора
   cur_client.execute ( "SELECT * FROM users WHERE
   username =? ", username)
   # Пошук імені користувача в базі
   check = cur_client.fetchone ()
   # Отримання рядки результатів
   conn_client.close ()
   # Закриття з'єднання
   if check! = None:
   # Якщо ім'я користувача вже існує return False
   else:
   return True
   # Продовження реєстрації при успішному проходженні перевірки
   # Отримання інших реєстраційних даних від клієнта
   address = api.getnewaddress ()
   # Отримання blockchain-адреси
   # Для користувача
   insert = (username, pass_hash, address, first_name, last_name, birthday) #
формування набору даних для занесення в базу даних
   # Підключення до бази даних, створення курсора
   cur_client.execute ( "INSERT INTO users VALUES (?,?,?,?,?,?)", insert)
   # Додавання даних користувачів в базу
   conn_client.commit ()
   # Підтвердження додавання запису
   conn_client.close ()
   # Закриття з'єднання#на стороне клиента
```

3.4.3 Python

Python (Пітон) - високорівнева неспеціалізований мову програмування, найчастіше використовується в якості скрипт-мови разом з відповідним інтерпретатором, що робить виконання коду трохи повільніше, ніж у компільованих мов, однак спрощує налагодження. Головна мета дизайну Python - легко читається вихідний код і підвищення продуктивності програміста. Python має простий синтаксисом і використовує відступи в якості головного структурирующего елемента. Незважаючи на деяку мінімалістичний синтаксису ядра, Python є дуже гнучким і широко застосовним за рахунок стандартної бібліотеки, що включає в себе широкий набір корисних функцій. Крім того, дана мова програмування підтримує кілька парадигм програмування (об'єктно і аспектно-орієнтоване, структурний, функціональний та імперативне). Архітектурними особливостями цієї мови є:

- динамічна типізація (не потрібно визначати тип змінної в момент оголошення, змінна зв'язується з ним при присвоєнні їй значення);
- повна інтроспекція (тип і структура об'єкта може бути визначена безпосередньо під час виконання програми);
- автоматичне керування пам'яттю (збірка сміття, відсутність витоків);
- механізм обробки виключень;
- підтримка багатопоточних обчислень.

Крім цього, у Python є ще деякі приємні особливості, такі як інтеграція C / C ++, вбудована підтримка Unicode в рядках, кроссплатформенность (відмінності у функціонуванні програм можуть виникнути лише в рідкісних випадках, але їх легко передбачити завдяки наявності докладної документації).

Вибір на користь даного мови програмування для написання прототипу функціоналу системи був зроблений через гнучкості, відносну простоту вивчення та можливості використання для вирішення різних завдань, а також наявності безліч бібліотек, що дозволяють працювати з іншими технологіями, задіяними в проекті.

3.4.4 База данных SQLite

SQLite - це вбудована в процес бібліотека, яка реалізує автономний, бессерверной, що не вимагає спеціальної попередньої настройки, транзакційний SQL-движок. Код SQLite переданий в суспільне надбання і, отже, може бути безоплатно використаний будь-якою метою, як комерційних, так і особистих. SQLite - сама широко поширена база даних в світі, яка використовується в незліченній кількості додатків, включаючи кілька гучних проектів (Google Chrome, Safari, Garena, Adobe Photoshop Lightroom і ін.).

Слово "вбудовується" в даному контексті означає, що SQLite не використовує парадигму клієнт-сервер і, отже, не має окремого серверного процесу (і взагалі кажучи, не є окремим працюючим процесом, а надає програмі бібліотеку, з якої та компонується, і движок бази даних стає частиною програми). SQL-база даних цілком зберігається в одному файлі (всі таблиці, індекси і т.д.). Формат файлу бази даних є

кросплатформним, база може бути безперешкодно скопійована або перенесена, наприклад, з 32-бітної системи на 64-бітну. Ця особливість робить SQLite широко поширеним вибором в якості формату файлу програми (файл, який використовується для збереження стану програми на диску або для обміну інформацією між додатками).

SQLite добре показує себе в таких випадках використання: • вбудовується обладнання та Інтернет речей;

- формат файлу програми;
- веб сайти;
- аналіз даних;
- кеш для корпоративних даних;
- база даних на сервері;
- файлові архіви;
- заміна ad hoc файлів на диску;
- внутрішні або тимчасові бази даних;
- заміна корпоративної бази даних в демо-версії і при тестуванні;
- освіту і навчання;
- експериментальні розширення мови SQL.

Читати дані з бази можуть кілька процесів або потоків одночасно, але запис можна здійснити лише за умови, що ніяких інших запитів в даний момент не обслуговується. Інакше спроба запису буде невдалою, в програму буде повернуто код помилки. Як альтернативний варіант, можна повторювати спроби записи протягом заданого проміжку часу.

Бібліотека SQLite дуже компактна - її розмір з повним набором функцій, в залежності від цільової платформи і налаштувань оптимізації компілятора, може бути менше 500 Кб. Якщо опціональні функції опущені, розмір бібліотеки SQLite може бути зменшений до менш ніж 300 Кб. SQLite може бути адаптована для роботи в мінімальному просторі стека (4 Кб) і в дуже маленькій "купі" (heap, 100 Кб), що робить її популярним движком БД для пристроїв з обмеженими можливостями пам'яті, таких як телефони і MP3-плеєри. Існує компроміс між використанням пам'яті і швидкодією. SQLite зазвичай працює тим швидше, чим більше пам'яті їй виділяється. Проте, продуктивність, як правило, досить хороша навіть в середовищах з низькою пам'яттю.

Сама бібліотека написана на мові C, але існує безліч прив'язок до інших мов програмування (C ++, Java, Python, Perl і багатьом іншим, повний список розміщений на інтернет-сторінці проекту).

SQLite, за твердженням розробників, дуже ретельно тестується перед кожним новим випуском і має репутацію надійного рішення. Автоматизований набір тестів реалізує величезну кількість тестових сценаріїв, що включають в себе мільйони індивідуальних SQL-запитів, і досягає 100% покриття гілок тестами. SQLite грамотно реагує на помилки виділення пам'яті і помилки введення-виведення. Транзакції зберігають ACID-властивості, навіть якщо вони були перервані через збої системи або харчування, що підтверджено тестуванням з імітацією збоїв системи. Якщо при експлуатації все ж виявляються помилки, вони публікуються в списках помилок.

Код SQLite підтримується інтернаціональною командою розробників, які продовжують розширювати можливості і підвищувати надійність і продуктивність даного продукту. Хоча вихідний код поставляється вільно, професійна підтримка також доступна.

4. ВИЯВЛЕННЯ ВІДПОВІДНОСТЕЙ НА ОСНОВІ ВИЯВЛЕНОЇ ФУНКЦІЇ

Типовий алгоритм зіставлення відбитків знаходить ступінь подібності між двома зображеннями. Цей ступінь подібності можна виміряти або шляхом рішення (одне для прийняття і нуль для відхилення) або за рахунком (від нуля до одного). Більшість методів відповідності порівнює різні функції. Проте деякі з відповідних алгоритмів безпосередньо використовують зображення відбитків пальців. Алгоритми відповідності придатні для повного зображення відбитків пальців. Звідси виникає складність у порівнянні відбитків пальців. Копіювання відбитків пальців можна розділити між трьома категоріями, а саме: узгодження на основі кореляції, узгодження на основі мініатюри та підбір відповідності на основі хребта. У цій дисертації використовується функція, засновану на узгодженні разом з мінімума. Три загальноприйняті методи відповідності пояснюються наступним чином.

• Відповідність на основі кореляції: кореляція між двома відбитками пальців виявляється для різного вирівнювання. Нехай I та T вводять зображення та відбиток зображення у шаблоні, відповідно. Сума різниці (SSD) є мірою подібності між двома зображеннями. Для одного зображення SSD дорівнює нулю. Більше різниця між двома зображеннями, більше значення SSD. SSD оцінюється як,

$$SSD(T, I) = "T - I" = (T - I)^T (T - I) = "T"^2 + "I"^2 - 2T^T I$$

$T^T I$ - кореляція між зображеннями T та I. Оскільки $"T"^2$ і $"I"^2$ постійні, SSD залежить від кореляції. Менша кореляційна цінність, більше подібність. Внутрішня проблема виникає під час здійснення перекладеного та повернутого зображення відбитків пальців. Розглянемо I (Δx, Δy, θ) зміщений та обернений вхід,

$$CC(T, I^{(\Delta x,\Delta y,\theta)}) = T^T I^{(\Delta x,\Delta y,\theta)}$$

Для збільшення відповідної оцінки кореляція максимізується, а баланс відповідності визначається як:

$$M(T, I) = max\Delta x,\Delta y,\theta CC(T, I^{(\Delta x,\Delta y,\theta)})$$

На практиці, кореляція на основі узгодження не підходить через нелінійні спотворення, стан шкіри, різниця тиску відбитків пальців, а також обчислювальні витрати

- Відповідність на основі мініатютюр: відповідність на основі мініатюри є найпопулярнішим підходом для збігу відбитків пальців. У цій техніці відбиток представлений вектором властивості, який містить мінуси. Кожен мінімум можна описати за місцем розташування, орієнтацією та типом мінусу. Проте більшість поточних алгоритмів відповідності використовують розташування та орієнтацію мініатюри і можуть бути представлені у вигляді m = x, y, θ, де θ - орієнтація, x та y - X-координати та Y-координати відповідно. Нехай T та I представляють набір мініатюр для шаблону та вхідного зображення, виражене як,

$$T = \{m_1, m_2, m_3, ..., m_m\}, m_i = \{x_i, y_i, \theta_i\}, i = 1..m$$

$$I = \{m', m', m', ..., m'\}, m = \{x', y_j, \theta'\}, j = 1..n$$

m та n - кількість мінімумів у шаблоні вхідного відбитка пальця.

Реєстрація здійснюється для вирівнювання вхідного зображення з зображенням шаблону.

- Відповідність функції "хребет": існує багато причин для розгляду більшої кількості функцій, крім «мінумумів». Дуже складно видобути мінуси з неякісного зображення відбитків пальців. Методи порівняння на основі протоколу не відповідають, коли на відбитку пальця міститься менше 40 цифр мініатюри. Щоб уникнути цих випадків, використовуються більш докладні функції. Розглядання додаткових функцій разом з мінусами збільшить надійність та точність.

У цій тезі, пори, точки та початкові хребти використовуються як функції для відбитка пальців. Для підвищення точності відповідності використовуються різні методи злиття. У функціональному рівні «fusion»,

декілька функцій надається відповідному алгоритму, і одержується одна відповідна оцінка. При збігу рівней, кожна функція підбирається самостійно і отримується оцінка. Тоді загальні результати відповідності злиття, використовуючи метод зварювання збірок, як правило формують суми та методи для створення ядра. «Fusion»- це вдосконалена версія збігу результатів, в якій результати матчів зливаються на основі певного правила злиття. У запропонованій роботі функція рівня 2 і рівня 3 була сформована з використанням рівня зварювання. Для узгодження використовується триангуляція Делоне. Пори також розглядаються для узгодження.

4.1 Пропоновані точки та початкове узгодження за допомогою триангуляції Делоне

Перед тим як узгодити вхідні мінімуми набору P (всі функції, включаючи функції рівня 2 і рівня 3) повинні бути відкалібровані відповідно до шаблону мініатюри, встановленої в межах тієї ж системи координат, шляхом обчислення перекладу, обертання та деформації вхідного відбитка пальця щодо відбитка пальця занесеного в базу. Для розрахунку цих з мінних необхідна референтна точка. Одним з найпростіших методів для контрольної точки є створення так називаємої «особливої» точки. Але відбиток пальців наврядчи містить особливу точку. $P' \subset P$ та $T' \subset T$ - це набори точок, що містять лише функції 2 рівня вхідного і шаблонного відбитків пальців відповідно. Для розрахунку контрольної точки використовується аналогічний векторний трикутник. Трикутник Делоне побудований за допомогою витягнутих точок рівня 2. Для розрахунку параметра калібрування знайдено пару відповідних трикутників. У роботі було порівняно трикутну пару, поки не було отримано відповідну пару. Параметри калібрування розраховуються з підходящої трикутної пари. Для розрахунку цих параметрів використовуються деформації краю та куту відповідної трикутної пари.

Ребро деформації:

$$E1 = eA1/eB1$$
$$E2 = eA2/eB2$$

$$E3 = eA3/eB3$$

Деформація обертання:

$$A1 = (A\alpha 1 - A\beta 1) - (A\alpha 2 - A\beta 2)$$
$$A2 = (A\beta 1 - A\gamma 1) - (A\beta 2 - A\gamma 2)$$
$$A3 = (A\gamma 1 - A\alpha 1) - (A\gamma 2 - A\alpha 2)$$

де *eA1*, *eA2* та, *eA3* - ребра трикутника з вхідних мінімальних множин. *eB1*

eB2, *eB3* є краями трикутника з набору мікросхем шаблону. Аналогічно, *Aα1, Aβ1* та *Aγ1* - кути трикутника з вхідних мінусових множин. *Aα2, Aβ2* та *Aγ2* - кути трикутника з безлічі шаблонів.

$$|E1 - E2| < th1$$

$$|E2 - E3| < th2$$

$$|E3 - E1| < th3$$

$$|A1 - A2| < th4$$

$$|A2 - A3| < th5$$

$$|A3 - A1| < th6$$

Після позиціонування контрольної точки було проведено обчислення відбитків пальців на переклад, обертання та деформацію, щоб провести калібрування обох відбитків пальців в тій же системі координат.

Параметр деформації:

$$\text{scale\_rate} = \frac{E1 + E2 + E3}{3.0}$$

Параметр напрямку обертання:

$$\text{rotate\_rate} = \frac{A1 + A2 + A3}{3.0}$$

Після чого було перетворено набір функцій вводу P у ту ж систему координат, що і шаблон T, за допомогою формули перетворення Hough та параметра вище.

Функції рівня 2 і рівня 3 були використані для повного узгодження. Завдяки унікальності, стабільності, структурній стійкості та лінійній складності часу, триангуляція Делоне дає дуже хороші результати при узгодженні. Триангуляція Делоне вже використовується з функціями рівня 2 для індексування та ідентифікації відбитків пальців. Однак, було використано триангуляцію Делоне не тільки для узгодження, але і для встановлення зв'язку між рівнями 2 і 3 рівня. З наборів мініатюр, трикутник Делоне формується за допомогою наступних кроків:

- З урахуванням точок n мініатюр, діаграма побудована таким чином, що кожен регіон діаграми буде містити одний мінумум.

- Триангуляція Делоне побудована шляхом приєднання мініатюри.

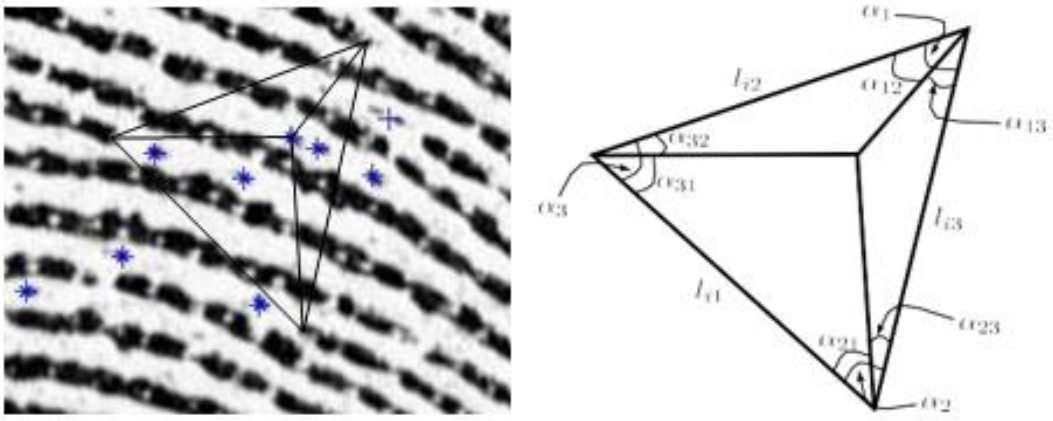
Алгоритм 1. Вибір трикутної пари

```
Require: A[1 : M], T [1 : N]
Ensure: Matched Array Index[1 : 2 × K] contains all the matching pair index of A
and T
1: Initialize: Matched Array Index[1 : 2 × K] = 0, j = 1 ,p = 1, K = M/3
2: for k = 1 to K do
3: if A[i] = T [j] A[i + 1] = T [j + 1] A[i + 2] = T [j + 2] then
4: Matched Array Index[p] = i
5: Matched Array Index[p + 1] = j
6: p = p + 2
7: i = i + 3
8: j = j + 3
9: else if A[i] > T [j] then
10:i = i + 3
11:else
12:j = j + 3
13:end if
14: end for
```

4.2 Поєднання дірок за допомогою алгоритму RAICP з двонаправленою відстані

Алгоритм Iterative Closest Point (ICP) є одним з кращих методів для задачі реєстрації набору точок. Алгоритм ICP ітеративно вирівнює два набори точок та обчислює вектор перекладу та обертання шляхом мінімізації квадратної відстані. Ці векторні перетворення використовуються для реконструкції одного набору точок по відношенню

до інших наборів точок. Цей алгоритм працює добре для жорсткої реєстрації, але не працює для не жорсткої реєстрації. Зображення відбитків пальців показують не жорстку властивість через зміну тиску, тоді як відбиток пальця приймається явно або неявно. Ця двонаправлена відстань також збігається до місцевого мінімуму. Крім того, для отримання найкращої відповідності між двома наборами точок, незалежний аналіз компонентів (ICA) використовується для оцінки вихідного параметра. Трикутник Делоне зображений на рисунку 4.1



(а) (б)

Рис. 4.1 - (а) – Трикутник Делоне побудований з використанням мінімуів, (б) - Особливості, витягнуті з трикутника.

4.3 Використання методів ідентифікаціх на проекті в реальному часі

На даний момент методи використовується для підвищення ефективності захисту інформації з використанням відбитків пальців на проекті LevelNet. LevelNet об'єднує практично всі існуючі антивірусні технології в єдин простий, інтуїтивно зрозумілий додаток. Ця розподілена система дозволяє її учасникам обмінюватися даними про небезпеки з будь-якої точки світу в режимі реального часу. LevelNet значно розширює можливості будь-якого користувача антивірусного програмного забезпечення. Якщо ж користувач не встановив на своєму пристрої ніяких додатків для захисту від вірусів і атак, LevelNet використовує можливості розподіленої мережевої системи і діє як самостійне

прикладне ПО, тому треба перевіряти вхідні дані кожного користувача за рахунок сщитування відбитків пальців.

Візуалізація працездатності програми на різних платформах:

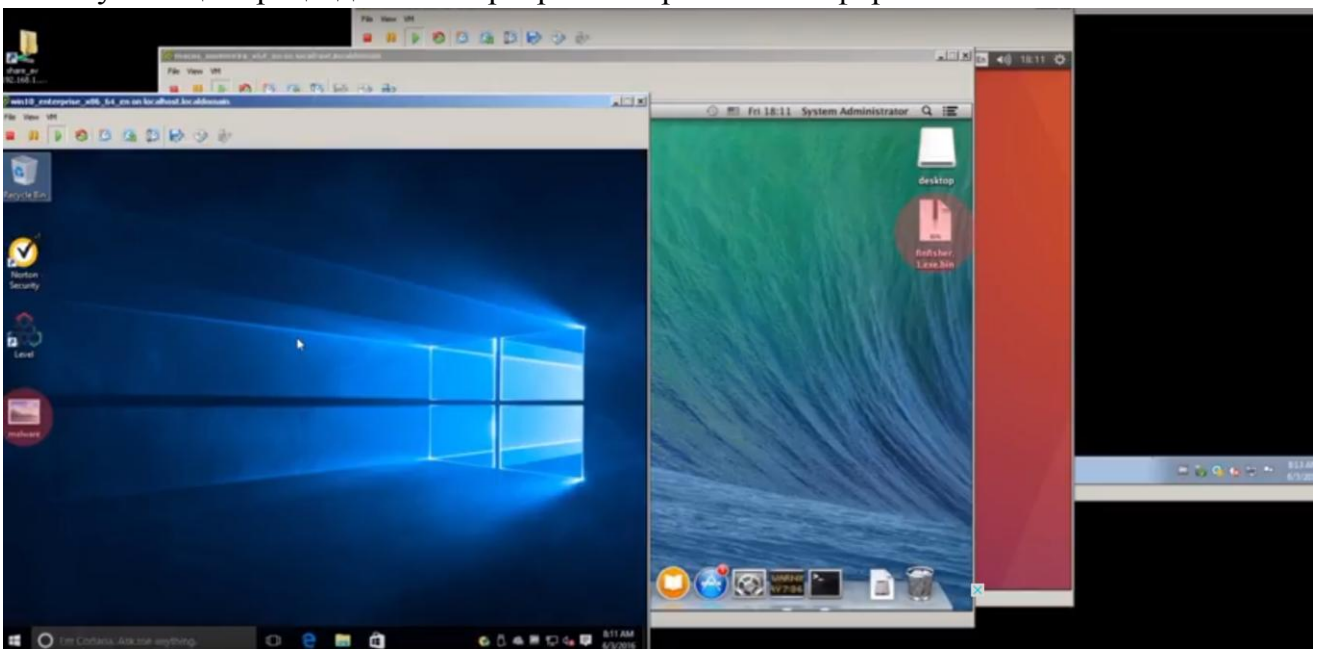


Рис. 4.2 - Візуалізація працездатності

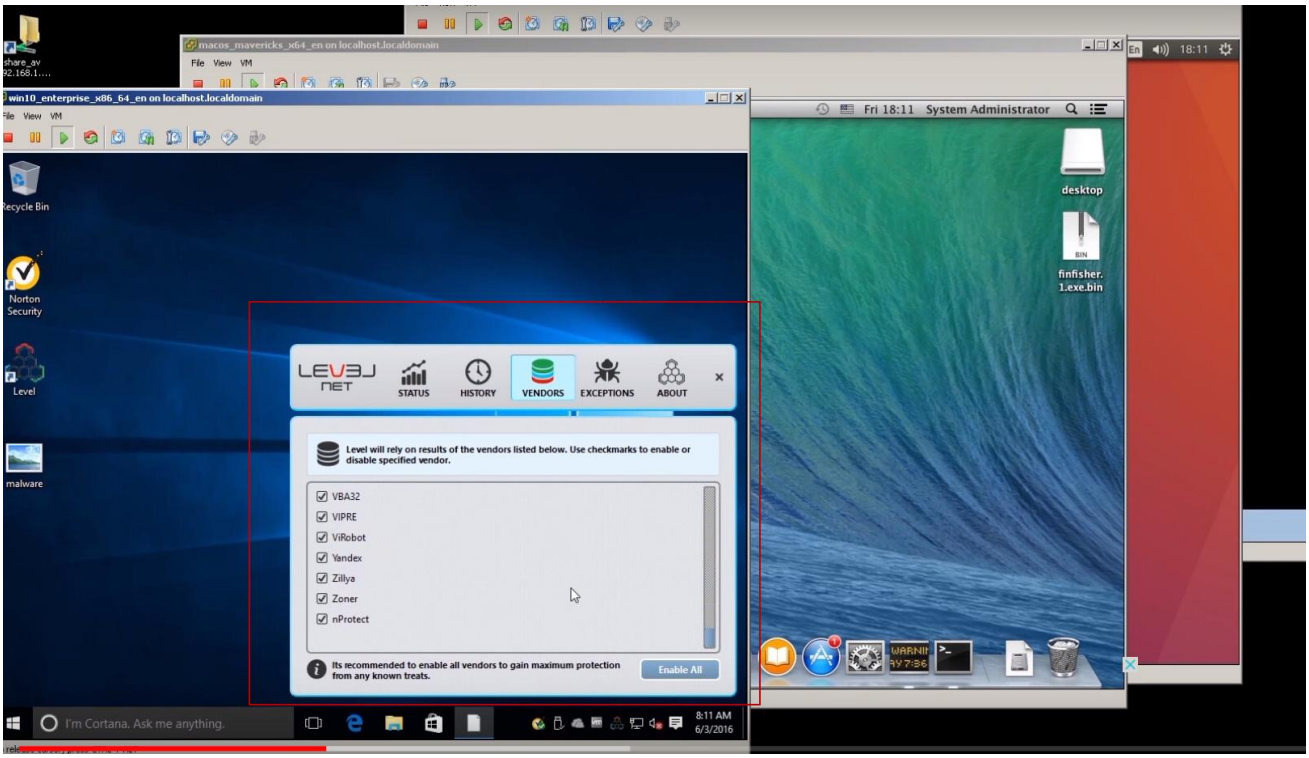


Рис. 4.3 - Візуалізація «постачальників»

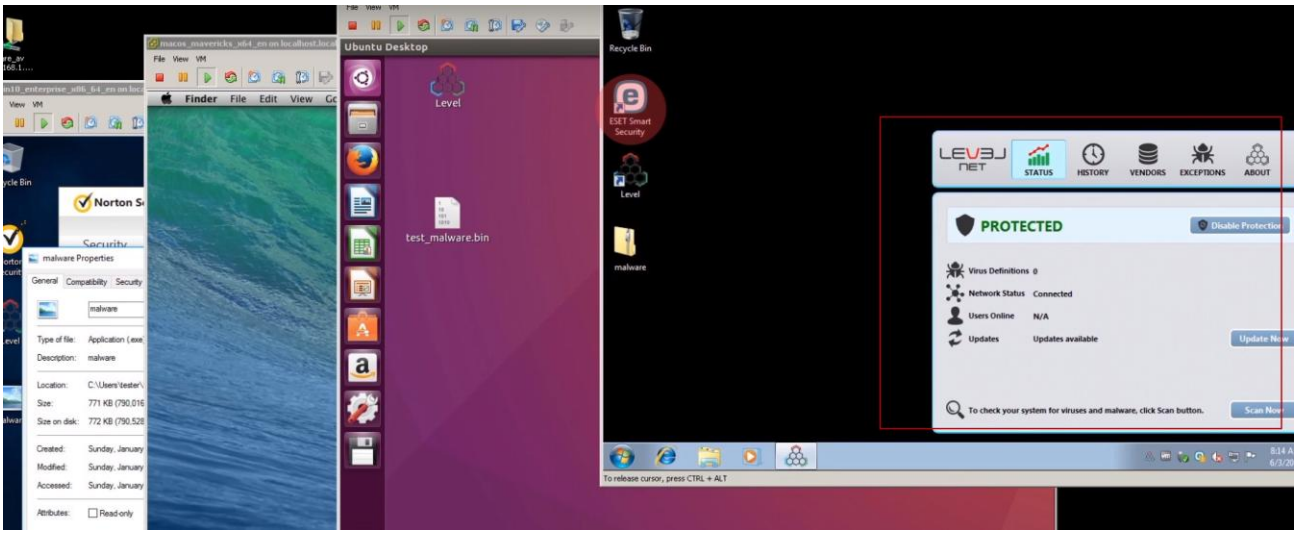


Рис. 4.4 - Статус перевірки

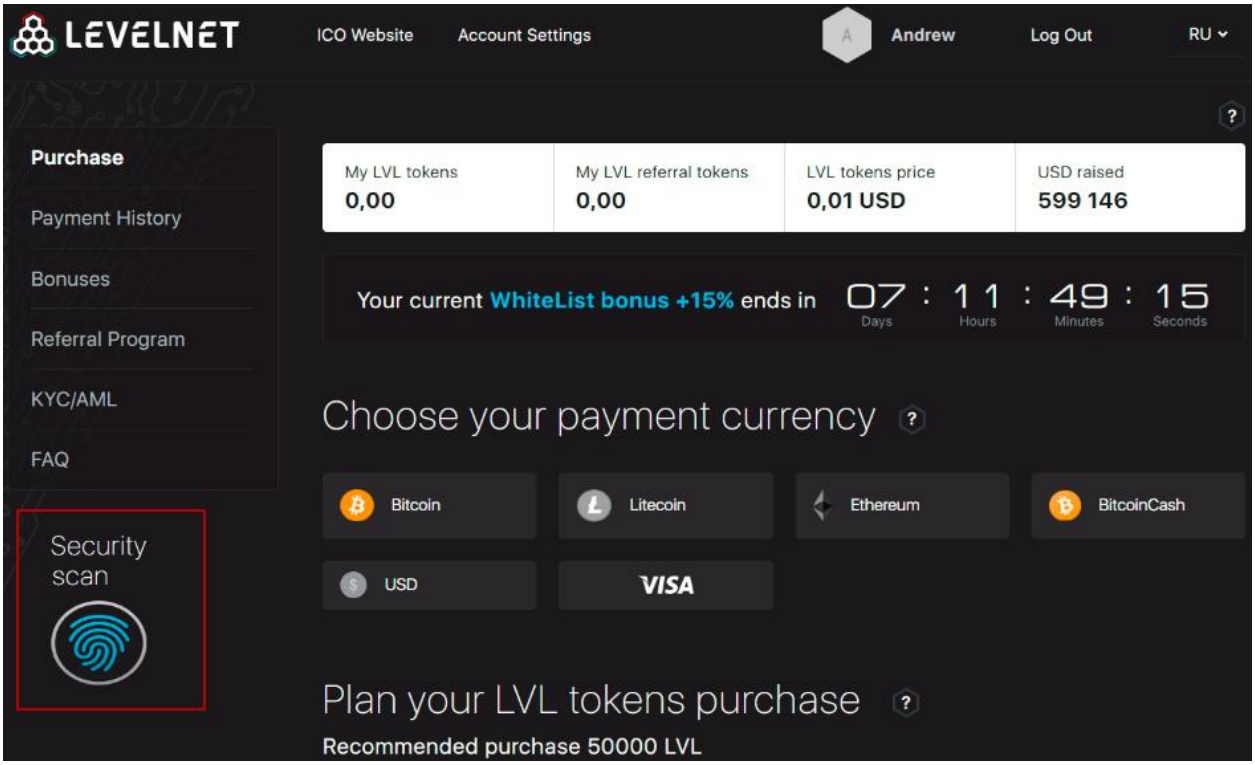


Рис. 4.5 - Передача даних для аналізу окремого користувача за допомогою відбитків пальців у персональному кабінеті

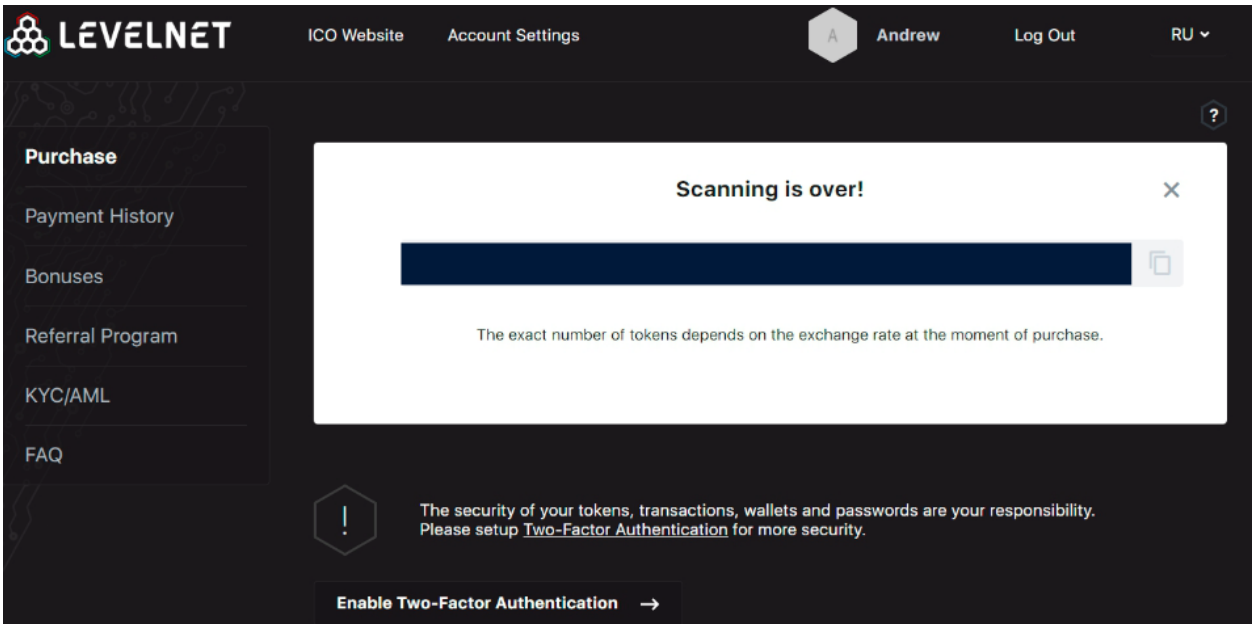


Рис. 4.6 - Процес сканування

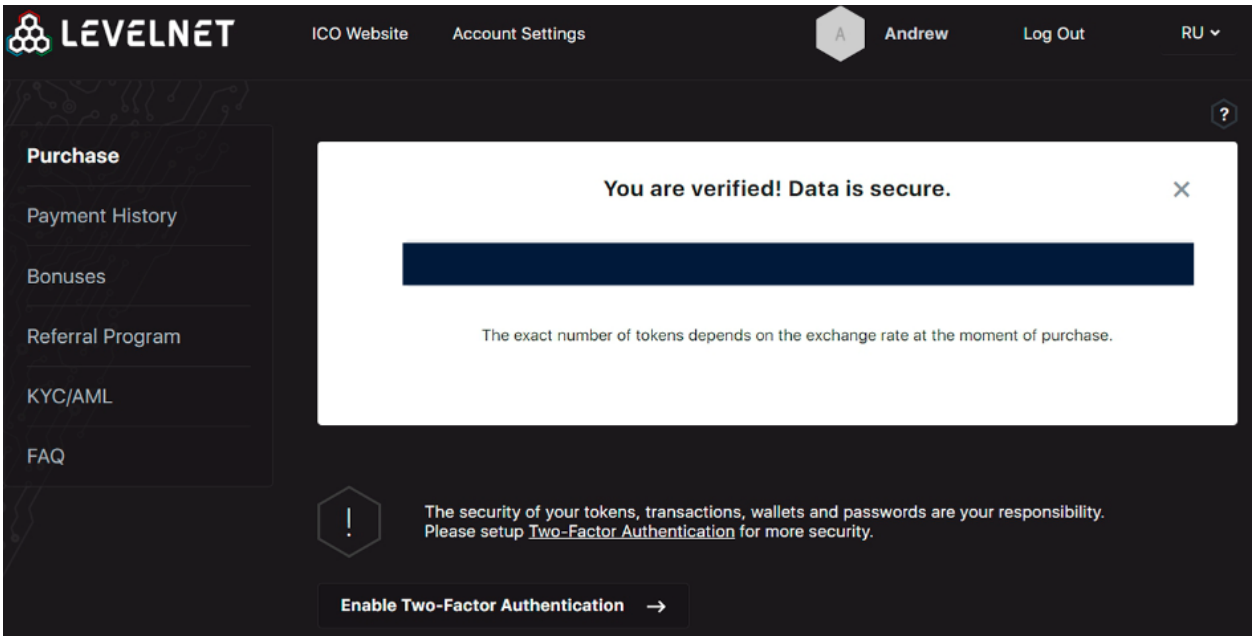


Рис. 4.7 - Результат вдалого сканування

ВИСНОВКИ

Вивчено і впроваджено симуляцію реальної системи автентифікації за відбитками пальців, такі як препроцесор відбитків пальців, екстрактор мінімумів та відповідність відбитків пальців. Попередні результати тестування з 40+ добровольцями були отримані та оброблені як очікувалося. Цей проект відкрив нові двері та дав зрозуміти, що системи автентифікації в реальному часі професійні, точні та складні. Елементарна індексація підходу групування людей з великої групи була включена. Для оцінки ефективності запропонованого алгоритму використовувались дві загальнодоступні бази даних: база даних NIST SD30 та база даних IIIT Delhi. Повна відповідність відбитків пальців виконана. Часткове друкування побудовано шляхом обрізання вікна з повного відбитка пальця двома способами: без перекриття та випадкового обрізання. У першому експерименті додавалися точки та початкові хребти з функцією рівня 2 за допомогою BOZORTH3 з NBIS. За базою даних NIST SD30 точність становить 88% і 90%. У той час як, над базою даних IIIT Delhi точність становить 86%. У другому експерименті використовується запропонована методика відповідності. Коефіцієнт підрахунку точок та інцидентів і функція рівня 2 розраховується за допомогою правила суми. За базою даних NIST SD30 точність становить 91% і 92%. В базі даних IIIT Delhi, точність становить 89% та 90% для неперекритого та випадкового обтинання відповідно. У третьому експерименті зіставлення балів точок-і-наповнювачів, пори та функції рівня 2 додано за допомогою правила суми. За допомогою бази даних NIST SD30, для неперекритого обтинання та випадкового обтинання алгоритм показує точність 93% та 95% відповідно. Проте над базою даних IIIT Delhi точність становить 94% та 92%. Результати досліджень,

викладені в цій дисертації, відкрили декілька напрямків досліджень, які дають змогу проводити подальші дослідження.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Майк МакГрайт Программирование на Python для начинающих : [перевод с англ. М.А. Райтмана] - М.: Эскмо, 2015. - 192 с.
2. Марк Саммерфилд Python на практике. / Пер. с англ. Слинкин А.А. - М.: ДМК Пресс, 2016. - 338 с.: ил.
3. JAIN A. K. AND PRABHKAR S. 2001. Fingerprint Matching Using Minutiae and Texture Features. Proceeding of International Conference on Image Processing (ICIP), pp. 282-285.
4. AGGARWAL G., RATHA N. K., TSAI-YANG J., AND BOLLE R. M. 2008. Gradient based textural characterization of fingerprints. In proceedings of IEEE International conference on Biometrics: Theory, Applications and Systems.
5. CHIKKERUR S., PANKANTI S., JEA A., AND BOLLE R. 2006. Fingerprint Representation using Localized Texture Features. The 18th International Conference on Pattern Recognition
6. JHAT Z. A., MIR A. H. AND RUBAB S. 2011. Fingerprint Texture Feature for Discrimination and Personal Verification. International Journal of Security and its Applications, Vol. 5, No. 3
7. V. Dixit, Deepti Singh, Parul Raj, M. Swathi, P. Gupta, kd-tree based fingerprint identification system 01/2008; DOI:10.1109/IWASID.2008.4688340
8. ZHENGU O., FENG J., SU F., AND CAI A., 2006. Fingerprint Matching with Rotation-Descriptor Texture Features. The 18th International Conference on Pattern Recognition, pp. 417-420.

9. Johan De Boer, Asker M Bazen, Sabih H Gerez, Indexing fingerprint databases based on multiple features Journal of The Acoustical Society of America- J ACOUST SOC AMER. 12/2001;

10. J.H. Wegstein, A Semi-automated Single Fingerprint Identification System. NBS Technical Note 481, April 1969.

11. J.H. Wegstein, The M40 Fingerprint Matcher. NBS Technical Note 878, July 1975.

12. R.T. Moore, Results of Fingerprint Image Quality Experiments. NBS Technical Report NBSIR 81-2298, June 1981.

13. R.T. Moore Automated Fingerprint Identification Systems - Benchmark Test of Relative Performance. American National Standard ANSI/IAI 1-1988, February 1988.

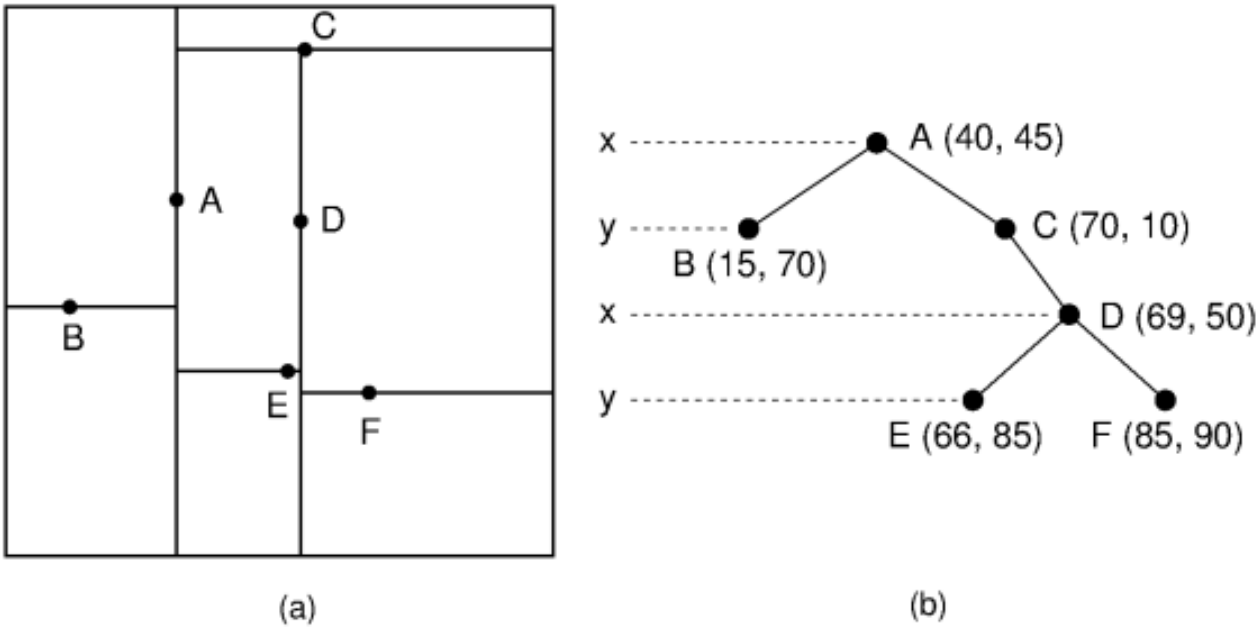
14. C. Wilson, M. Garris, C. Watson, A, Hicklin, Studies of Fingerprint Matching Using the NIST Verification Test Bed (VTB). Technical Report NISTIR 7020, July 2003.

15. C. Watson, C. Wilson, M. Indovina, R. Snelick, K. Marshall, Studies of One-toOne Matching with Vendor SDK Matchers. Technical Report NISTIR 7119, July 2004.

Додатки

Додаток 1
Копії графічних матеріалів

Схема KD-дерева
використаного у розробці



(a) (b)

Схема виявлення мініатюр
на відбитків

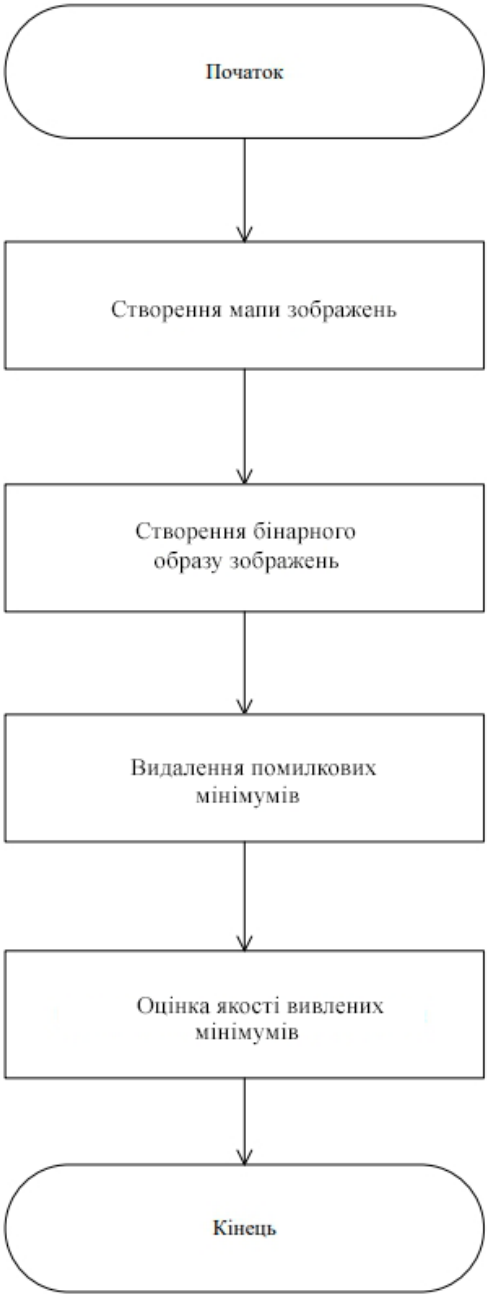


Схема операцій у blockchain

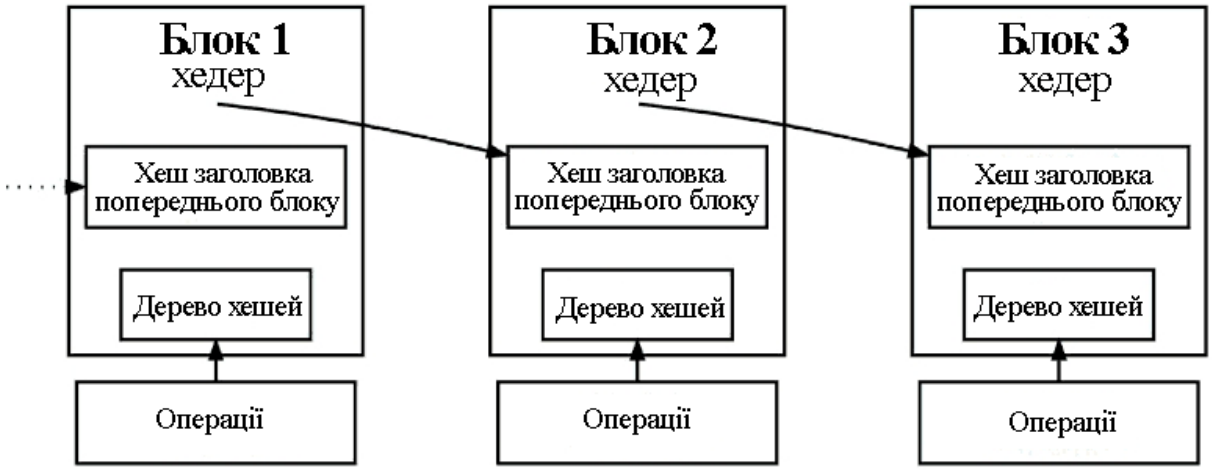
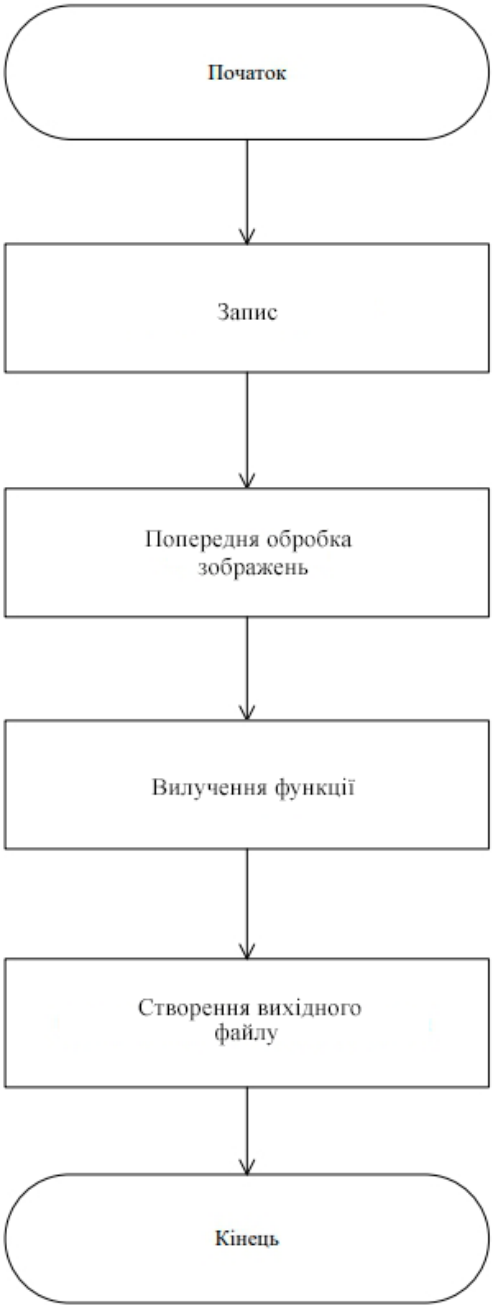


Схема реєстрації
відбитків пальців



Структурна схема градусного рівня синтезу

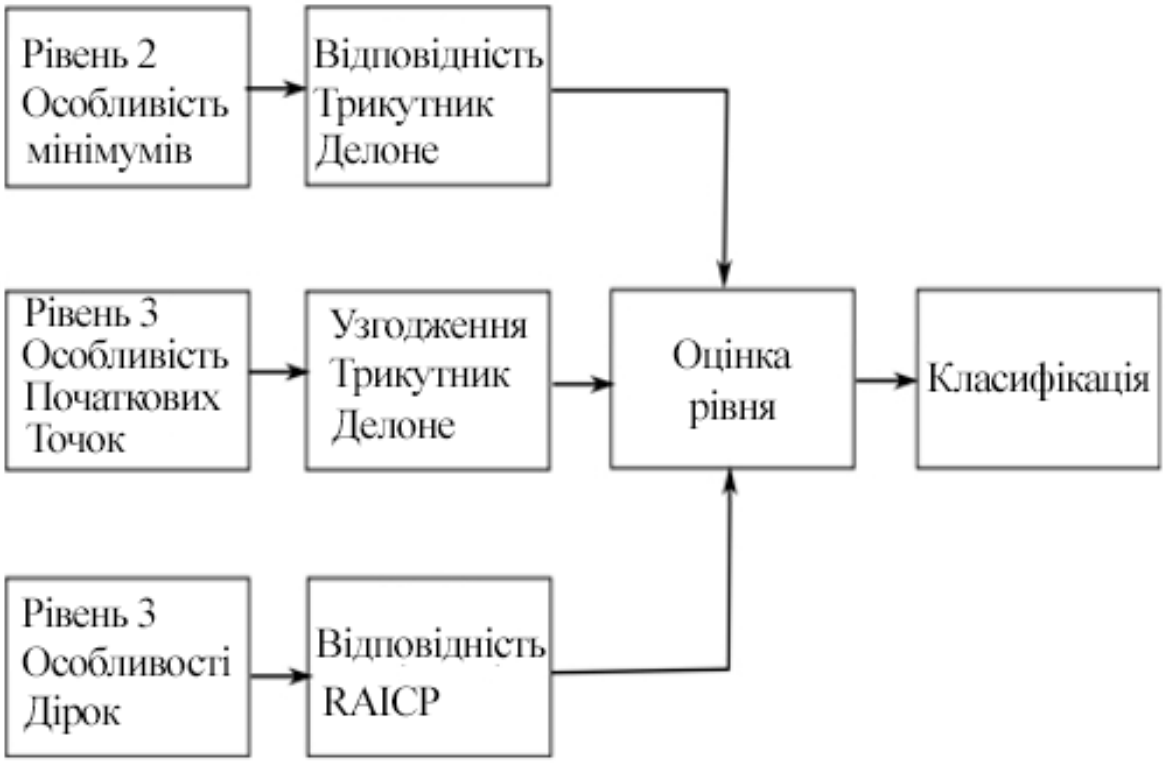


Схема проведення транзакцій
у проектах blockchain

Методи підвищення ефективності захисту інформації
з використанням відбитків пальців у blockchain проектах
Пацьора А.А.

Додаток 2
Копії публікацій по темі магістерської дисертації

УДК 004.441

Д.т.н., професор Зайцев В.Г., студент Пацьора А.А.
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

МЕТОДИ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ЗАХИСТУ ІНФОРМАЦІЇ З ВИКОРИСТАННЯМ ІДЕНТИФІКАЦІЇ ВІДБИТКІВ ПАЛЬЦІВ ДЛЯ ПРОЕКТІВ НА БЛОКЧЕЙНІ

Abstract

Volodymyr G. Zaitsev, assoc. prof., PhD; Andrii Patsora, student
Increasing efficiency methods of information protection with using fingerprint identification system for project on blockchain

This paper concerns the method of increasing efficiency methods of information protection with using fingerprint identification system for project on blockchain. The developed method is an adaptive identification method and is suitable for versatile applications. Using fingerprint identification in a block structure increases the truth and security of files in a chain using a scanner on any smartphone or using a connected scanner to computer.

Вступ

Блокчейн, тобто ланцюжок блоків транзакцій - розподілена база даних, яка підтримує перелік записів, так званих блоків, що постійно зростає. База захищена від підробки та переробки. Кожен блок містить часову мітку та посилання на попередній блок хеш дерева.

У процес розробки проекту входить вираховування криптографічного хешу відбитків пальців для кожного розрахованого блоку, котрий містить у собі записи. Хеш усієї структури, що містить відпечатки файлів, записується в «блок-ланцюжок» тим самим підвищує рівень достовірності даних. Правдивість цих файлів може бути незалежно підтверджена будь-якою стороною, яка має доступ до блочного ланцюжку

Постановка задачі

Створити спосіб ефективного захисту інформації для проектів, котрі використовують блочну структуру з використанням ідентифікації відбитків пальців на будь яких етапах запису інформації.

Провести експерименти за алгоритмами обраного способу по ідентифікації різних типів відбитків пальців на основі шаблонів хребтів і борозд на поверхні кінчиків пальців. Удосконалити первинну обробку для покращення контрастності зображення та зменшення помилок при вторинній обробці

Термінологія

Блокчейн – (block – блоки, chain – це ланцюжок) – це ланцюжок блоків. Іншими словами – це розподілена база даних, яка підтримує перелік записів, об'єднаних у блоки, що постійно зростає

Хеш – структура даних (хеш-таблиця), варіант реалізації асоціативного масиву.

Дерево KD - це структура даних про розподіл просторів для організації точок в k-просторі

Опис способу

Розроблений спосіб є адаптивним для ідентифікації та підходить для різностороннього застосування. Використання ідентифікації відбитків пальців у блочній структурі збільшує правдивість та захищеність файлів у ланцюжку з використанням сканеру у будь-якому смартфоні чи за допомогою підключеного сканеру до комп'ютера.

KD дерева є корисною структурою даних для пошуку багатовимірного ключа. На рисунку 1 показано приклад KD-дерева, котре використовується в роботі по ідентифікації відбитків пальців для блокчейн проектів.

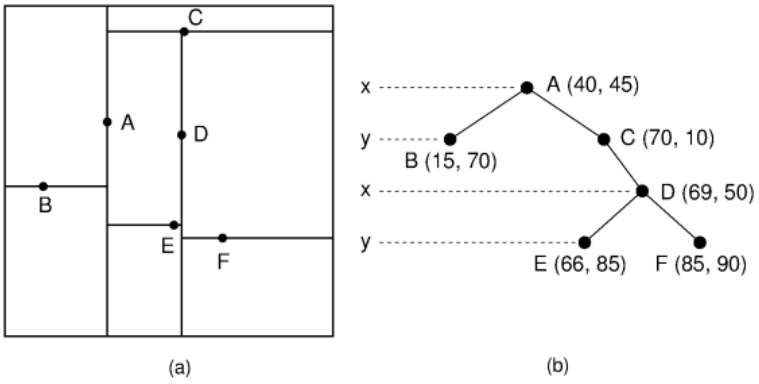


Рис. 1. Приклад KD-дерева використаного у розробці

Операції на KD-деревах: Insert (додавання), Delete (видалення), FindMin (пошук мінімального), Search (пошук) та Nearest Neighbor (пошук найближчого сусіда):

- Insert (створити, додати) – елементи в KD дереві перетинаються, щоб знайти відповідний лист, починаючи з кореня залежно від того, чи точка, яку потрібно вставити, знаходиться на "Лівій" або "Правій" стороні для розміщення нового запису.
- Delete (видалити) – щоб видалити точку з існуючого дерева KD, потрібно сформувати набір всіх вузлів і листів від молодших елементів (дітей) цільового вузла та відтворити цю частину дерева.
- FindMin (пошук мінімального) – за допомогою FindMin з'являється можливість знайти точку з найменшим значенням у параметрі dth.
- Search (пошук) – пошук елементів к KD дереві використовується за допомогою наступного алгоритма:

```
Input: T: Tree node, P: Point of interest
Output: T: Tree node

procedure Search(T, P)
  for all l such that T! = null do
        if T.data equals P then
             return null
        else if T.data[l] > P[l] then
             T ← T.lef t
        else
             T ← T.right
        end if
       l ← (l + 1) mod D
  end for
  return null
end procedure
```

- Nearest Neighbor (пошук найближчого сусіда) – алгоритм пошуку найближчого сусіда (NN) має на меті знайти точку в дереві, що найближча до заданої точки. Цей пошук можна зробити ефективно за допомогою властивостей KD-дерева для швидкого усунення великих частин пошукового простору.

В ході роботи були розроблені та встановлені мінімальні властивості обробки скану відбитку. На міліметр простору виділяється 12700 пікселей, котрі налаштовані на квантування до 256 рівнів сірого кольору. Програма може зчитувати файли у форматі: JPEGB[1], JPEGL[2] та ANSI/NIST[3]. Основна функція розробленої програми: вводить зображення, автоматично виявляє часткові зображення (мікрофільми) для сканування на відбиту.

Послідовність аналізу та оцінки ефективності створеного способу вираховування криптографічного хешу відбитків пальців для кожного розрахованого блоку:

1. Додавання відсканованого файлу з відбитком пальця у форматах описаних вище;
2. Створення карти зображення;
3. Створення бінарного образу;
4. Повторне сканування для виявлення дефектів зображення;
5. Виявлення та усунення дефектів;
6. Підрахування та аналіз сусідніх хребтів;
7. Циклічний пошук мінімального значення бінарного образу;
8. Вивід результату у файл.

Важливо відзначити два основні моменти стосовно датчика відбитків пальців:

- Основна функція сканування відбитку пальця обмежується розташуванням (x, y) та орієнтацією t, представленими як x, y, t;
- Розроблений алгоритм використовується для отримання інваріантності обертання та перекладу.

Функція алгоритму:

- Побудова таблиці порівняння внутрішніх відбитків пальці;
- Побудова таблиці сумісності між хребтами і борознами на поверхні кінчиків пальців;
- Створення перетинання у таблиці по KD дереву за допомогою об’єднання сумісних кластерів та накопиченню відповідностей у схемі.

Висновки

Розроблено спосіб створений для ідентифікації відбитків пальців і захоплення необхідних масивів. Алгоритм використовується на проекті у реальному часі для забезпечення високого рівня достовірності розпізнавання.

Алгоритм працює у середньому на 57% швидше аналогів, як показали дослідження, результати яких наведені нижче. Середній час сканування відбитку становить 2,4мс, що дуже важливо для ідентифікації на кожному рівні перевірки у блокчейн структурі.

A001.AN2.xyt	18:17:10.512	18:17:10.612	0:00:00.100	Y
A002.AN2.xyt	18:17:11.720	18:17:11.722	0:00:00.002	Y
A003.AN2.xyt	18:17:12.618	18:17:12.620	0:00:00.002	Y
A004.AN2.xyt	18:17:13.317	18:17:13.319	0:00:00.002	Y
A005.AN2.xyt	18:17:14.089	18:17:14.091	0:00:00.002	Y
A006.AN2.xyt	18:17:14.717	18:17:14.719	0:00:00.002	Y
A007.AN2.xyt	18:17:15.568	18:17:15.570	0:00:00.002	Y
A008.AN2.xyt	18:17:16.898	18:17:16.900	0:00:00.002	Y
A009.AN2.xyt	18:17:17.631	18:17:17.633	0:00:00.002	Y
A010.AN2.xyt	18:17:18.932	18:17:18.934	0:00:00.002	Y
A011.AN2.xyt	18:17:19.612	18:17:19.614	0:00:00.002	Y
A012.AN2.xyt	18:17:20.346	18:17:20.347	0:00:00.001	Y
A013.AN2.xyt	18:17:21.424	18:17:21.426	0:00:00.002	Y
A014.AN2.xyt	18:17:22.351	18:17:22.353	0:00:00.002	Y
A015.AN2.xyt	18:17:23.405	18:17:23.407	0:00:00.002	Y
A016.AN2.xyt	18:17:24.442	18:17:24.443	0:00:00.001	Y
A017.AN2.xyt	18:17:25.343	18:17:25.345	0:00:00.002	Y
A018.AN2.xyt	18:17:26.346	18:17:26.348	0:00:00.002	Y
A019.AN2.xyt	18:17:27.485	18:17:27.487	0:00:00.002	Y
A020.AN2.xyt	18:17:28.332	18:17:28.334	0:00:00.002	Y
A021.AN2.xyt	18:17:29.293	18:17:29.295	0:00:00.002	Y
A022.AN2.xyt	18:17:30.024	18:17:30.026	0:00:00.002	Y
A023.AN2.xyt	18:17:30.733	18:17:30.735	0:00:00.002	Y
A024.AN2.xyt	18:17:31.581	18:17:31.583	0:00:00.002	Y
A025.AN2.xyt	18:17:32.851	18:17:32.854	0:00:00.003	Y
A026.AN2.xyt	18:17:34.241	18:17:34.243	0:00:00.002	Y
A027.AN2.xyt	18:17:35.055	18:17:35.057	0:00:00.002	Y
A028.AN2.xyt	18:17:36.963	18:17:36.964	0:00:00.001	Y
A029.AN2.xyt	18:17:37.808	18:17:37.810	0:00:00.002	Y
A030.AN2.xyt	18:17:39.490	18:17:39.492	0:00:00.002	Y
A031.AN2.xyt	18:17:40.232	18:17:40.234	0:00:00.002	Y
A032.AN2.xyt	18:17:41.239	18:17:41.241	0:00:00.002	Y
A033.AN2.xyt	18:17:41.932	18:17:41.934	0:00:00.002	Y
A034.AN2.xyt	18:17:42.977	18:17:42.979	0:00:00.002	Y
A035.AN2.xyt	18:17:44.170	18:17:44.172	0:00:00.002	Y
A036.AN2.xyt	18:17:45.552	18:17:45.554	0:00:00.002	Y
A037.AN2.xyt	18:17:46.723	18:17:46.725	0:00:00.002	Y
A038.AN2.xyt	18:17:47.582	18:17:47.584	0:00:00.002	Y
A039.AN2.xyt	18:17:47.995	18:17:47.997	0:00:00.002	Y
A040.AN2.xyt	18:17:48.803	18:17:48.805	0:00:00.002	Y
A041.AN2.xyt	18:17:49.295	18:17:49.297	0:00:00.002	Y
A042.AN2.xyt	18:17:49.659	18:17:49.661	0:00:00.002	Y
A043.AN2.xyt	18:17:49.938	18:17:49.940	0:00:00.002	Y
A044.AN2.xyt	18:17:50.573	18:17:50.575	0:00:00.002	Y

Рис. 2. Тестування аналізу відбитків пальців та пошук відповідностей у базі даних

Література

1. Джамшед Мавалвала, История иследования отпечатка пальца, http://magichand.ru/all-article-encyclopedias/.../899-hd
2. P. Cube. Hash Pover–A., Podcast: http://investorfieldguide.com/hashpower/
3. Мелани Свон, Схема новой экономики на Блокчейне, страниц: 240, 2016.

УДК 004.441

Д.т.н., професор Зайцев В.Г., студент Пацьора А.А.

Інформаційний портал http://www.securitynewspaper.com

АВТЕНТИФІКАЦІЯ ВІДБИТКУ ПАЛЬЦІВ ТА ІНТЕГРАЦІЯ У BLOCKCHAIN
AUTOMATIC FASTENING OPENING AND BLOCKCHAIN INTEGRATION

At the stage of building the architecture of the system based on the previously defined requirements and taking into account the selected technologies, diagrams of use options, components, activities and consistency were developed. The diagrams in various aspects illustrate the composition and behavior of the systems being

developed. This section covers only a few examples of different types of charts, a complete list of diagrams in the application.

The client PC must have a preinstalled browser to access the application's web page. Also, on the Client's PC there is a copy of Blockchain, which automatically downloads during registration and updates when logged in.

The application page contains control elements that allow you to interact with the system and use its functionality by sending commands to the server.

The application server performs functions according to the received requests. The functional program is conventionally divided into 4 parts:

• Manage accounts (create a user account and rights owner, authenticate users and copyright holders in the system);

• Music management (recording of records (available only for rights holders), authentication of records);

• Metadata management (attaching and reading metadata);

• Purchase management (transaction implementation).

The ClientDB database contains data about registered users (Users table) and rightsholders (rights holders table), access to the database is carried out using SQL queries from the code part of the system server. The user data table contains the following fields:

- User name;
- Hash-value of the password;
- Name;
- Surname;
- Date of birth.

Client-server architecture is a computer network architecture in which many clients (remote processors) request and access services from the server. The client and server interact with each other through various network protocols such as IP protocol, HTTP protocol, FTP and others. Choosing a protocol for interaction depends on the scope of its application. Choosing a protocol for interaction depends on the scope of its application. The client computers provide an interface that allows the user of the computer to request access to the server services and display the results obtained from the server. The server waits for requests from customers and then responds to them. Ideally, the server provides a standardized, clear client interface to prevent the client from being aware of the specifics of the system (that is, hardware and software) that provides the service. The client part of the program is most often located on workstations or personal computers, while servers are located in other parts of the network, most often on more powerful machines.

This computing model is especially effective if clients and the server have clearly defined tasks that they usually perform. Many clients can access data on the server at the same time, and at the same time, client computers can perform other

actions. Since the server and client computers are considered to be smart devices, the client-server model is completely different from the old mainframe model in which the central mainframe computer performed all the tasks for the dummy terminals associated with it.

The marking of individual values stored in a table is presented as a search function in the given table. For example, the index of the lower left miniature is stored in the Pm table and is called k (Pm), while the distance between the two thumbnails is also stored in the Pm table and is called d (Pm). The lower right print is a print and uses similar notations, except that all of its pair comparisons were stored in the G table, and the sizes made on the two corresponding thumbnails in the gallery's print were stored in the Gn table. The following three tests are performed to determine whether the Pm and Gn records tables are compatible. The first test checks whether the respective distances are within the Td. The last two tests check whether they have relative angular minima within the given assumption T.

На рисунку 1 зображено підтвердження розміщення научної статті на сайті http://www.securitynewspaper.com

At the stage of building the architecture of the system based on the previously defined requirements and taking into account the selected technologies, diagrams of use options, components, activities and consistency were developed. The diagrams in various aspects illustrate the composition and behavior of the systems being developed. This section covers only a few examples of different types of charts, a complete list of diagrams in the application. The client PC must have a preinstalled browser to access the application's web page. Also, on the Client's PC there is a copy of Blockchain, which automatically downloads during registration and updates when logged in. The application page contains control elements that allow you to interact with the system and use its functionality by sending commands to the server. The application server performs functions according to the received requests. The functional program is conventionally divided into 4 parts: • Manage accounts (create a user account and rights owner, authenticate users and copyright holders in the system); • Music management (recording of records (available only for rights holders), authentication of records); • Metadata management (attaching and reading metadata); • Purchase management (transaction implementation). The ClientDB database contains data about registered users (Users table) and rightsholders (rights holders table), access to the database is carried out using SQL queries from the code part of the system server. The user data table contains the following fields: • User name; • Hash-value of the password; • Name; • Surname; • Date of birth. Client-server architecture is a computer network architecture in which many clients (remote processors) request and access services from the server. The client and server interact with each other through various network protocols such as IP protocol, HTTP protocol, FTP and others. Choosing a protocol for interaction depends on the scope of its application. Choosing a protocol for interaction depends on the scope of its application. The client computers provide an interface that allows the user of the computer to request access to the server services and display the results obtained from the server. The server waits for requests from customers and then responds to them. Ideally, the server provides a standardized, clear client interface to prevent the client from being aware of the specifics of the system (that is, hardware and software) that

Рис. 1 – Розміщена публікація на сайті

Додаток 3
Лістинг програми

ERC20Interface.json

```
{
        "contractName": "ERC20Interface",
        "abi": [
          {
            "constant": true,
            "inputs": [],
            "name": "symbol",
            "outputs": [
              {
                "name": "",
                "type": "string"
              }
            ],
            "payable": false,
            "stateMutability": "view",
            "type": "function"
```

```
},
{
  "anonymous": false,
  "inputs": [
    {
      "indexed": true,
      "name": "from",
      "type": "address"
    },
    {
      "indexed": true,
      "name": "to",
      "type": "address"
    },
    {
      "indexed": false,
      "name": "value",
      "type": "uint256"
    }
  ],
  "name": "Transfer",
  "type": "event"
},
{
  "anonymous": false,
  "inputs": [
    {
      "indexed": true,
      "name": "from",
      "type": "address"
    },
    {
      "indexed": true,
      "name": "spender",
      "type": "address"
    },
    {
      "indexed": false,
      "name": "value",
      "type": "uint256"
    }
  ],
  "name": "Approval",
  "type": "event"
},
{
  "constant": true,
```

```
  "inputs": [],
  "name": "decimals",
  "outputs": [
    {
      "name": "",
      "type": "uint8"
    }
  ],
  "payable": false,
  "stateMutability": "view",
  "type": "function"
},
{
  "constant": true,
  "inputs": [],
  "name": "totalSupply",
  "outputs": [
    {
      "name": "supply",
      "type": "uint256"
    }
  ],
  "payable": false,
  "stateMutability": "view",
  "type": "function"
},
{
  "constant": true,
  "inputs": [
    {
      "name": "_owner",
      "type": "address"
    }
  ],
  "name": "balanceOf",
  "outputs": [
    {
      "name": "balance",
      "type": "uint256"
    }
  ],
  "payable": false,
  "stateMutability": "view",
  "type": "function"
},
{
  "constant": false,
```

```
  "inputs": [
    {
      "name": "_to",
      "type": "address"
    },
    {
      "name": "_value",
      "type": "uint256"
    }
  ],
  "name": "transfer",
  "outputs": [
    {
      "name": "success",
      "type": "bool"
    }
  ],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function"
},
{
  "constant": false,
  "inputs": [
    {
      "name": "_from",
      "type": "address"
    },
    {
      "name": "_to",
      "type": "address"
    },
    {
      "name": "_value",
      "type": "uint256"
    }
  ],
  "name": "transferFrom",
  "outputs": [
    {
      "name": "success",
      "type": "bool"
    }
  ],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function"
```

```
},
{
  "constant": false,
  "inputs": [
    {
      "name": "_spender",
      "type": "address"
    },
    {
      "name": "_value",
      "type": "uint256"
    }
  ],
  "name": "approve",
  "outputs": [
    {
      "name": "success",
      "type": "bool"
    }
  ],
  "payable": false,
  "stateMutability": "nonpayable",
  "type": "function"
},
{
  "constant": true,
  "inputs": [
    {
      "name": "_owner",
      "type": "address"
    },
    {
      "name": "_spender",
      "type": "address"
    }
  ],
  "name": "allowance",
  "outputs": [
    {
      "name": "remaining",
      "type": "uint256"
    }
  ],
  "payable": false,
  "stateMutability": "view",
  "type": "function"
}
```

```
    ],
    "bytecode": "0x",
    "deployedBytecode": "0x",
    "sourceMap": "",
    "deployedSourceMap": "",
    "source": "pragma solidity ^0.4.11;\n\ncontract ERC20Interface
{\n     event Transfer(address indexed from, address indexed to,
uint256 value);\n     event Approval(address indexed from, address
indexed spender, uint256 value);\n     string public symbol;\n\n
function decimals() constant returns (uint8);\n       function
totalSupply() constant returns (uint256 supply);\n     function
balanceOf(address _owner) constant returns (uint256 balance);\n
function transfer(address _to, uint256 _value) returns (bool
success);\n     function transferFrom(address _from, address _to,
uint256 _value) returns (bool success);\n          function
approve(address _spender, uint256 _value) returns (bool success);\n
function allowance(address _owner, address _spender) constant
returns (uint256 remaining);\n}\n",
    "sourcePath":  "/Users/ahiatsevich/Development/blockchain/cav-
platform/contracts/ERC20Interface.sol",
    "ast": {
      "attributes": {
        "absolutePath":
"/Users/ahiatsevich/Development/blockchain/cav-
platform/contracts/ERC20Interface.sol",
        "exportedSymbols": {
          "ERC20Interface": [
            3269
          ]
        }
      },
      "children": [
        {
          "attributes": {
            "literals": [
              "solidity",
              "^",
              "0.4",
              ".11"
            ]
          },
          "id": 3195,
          "name": "PragmaDirective",
          "src": "0:24:7"
        },
        {
          "attributes": {
```

```
  "baseContracts": [
    null
  ],
  "contractDependencies": [
    null
  ],
  "contractKind": "contract",
  "documentation": null,
  "fullyImplemented": false,
  "linearizedBaseContracts": [
    3269
  ],
  "name": "ERC20Interface",
  "scope": 3270
},
"children": [
  {
    "attributes": {
      "anonymous": false,
      "name": "Transfer"
    },
    "children": [
      {
        "children": [
          {
            "attributes": {
              "constant": false,
              "indexed": true,
              "name": "from",
              "scope": 3203,
              "stateVariable": false,
              "storageLocation": "default",
              "type": "address",
              "value": null,
              "visibility": "internal"
            },
            "children": [
              {
                "attributes": {
                  "name": "address",
                  "type": "address"
                },
                "id": 3196,
                "name": "ElementaryTypeName",
                "src": "71:7:7"
              }
            ],
```

```
      "id": 3197,
      "name": "VariableDeclaration",
      "src": "71:20:7"
    },
    {
      "attributes": {
        "constant": false,
        "indexed": true,
        "name": "to",
        "scope": 3203,
        "stateVariable": false,
        "storageLocation": "default",
        "type": "address",
        "value": null,
        "visibility": "internal"
      },
      "children": [
        {
          "attributes": {
            "name": "address",
            "type": "address"
          },
          "id": 3198,
          "name": "ElementaryTypeName",
          "src": "93:7:7"
        }
      ],
      "id": 3199,
      "name": "VariableDeclaration",
      "src": "93:18:7"
    },
    {
      "attributes": {
        "constant": false,
        "indexed": false,
        "name": "value",
        "scope": 3203,
        "stateVariable": false,
        "storageLocation": "default",
        "type": "uint256",
        "value": null,
        "visibility": "internal"
      },
      "children": [
        {
          "attributes": {
            "name": "uint256",
```

```
                "type": "uint256"
              },
              "id": 3200,
              "name": "ElementaryTypeName",
              "src": "113:7:7"
            }
          ],
          "id": 3201,
          "name": "VariableDeclaration",
          "src": "113:13:7"
        }
      ],
      "id": 3202,
      "name": "ParameterList",
      "src": "70:57:7"
    }
  ],
  "id": 3203,
  "name": "EventDefinition",
  "src": "56:72:7"
},
{
  "attributes": {
    "anonymous": false,
    "name": "Approval"
  },
  "children": [
    {
      "children": [
        {
          "attributes": {
            "constant": false,
            "indexed": true,
            "name": "from",
            "scope": 3211,
            "stateVariable": false,
            "storageLocation": "default",
            "type": "address",
            "value": null,
            "visibility": "internal"
          },
          "children": [
            {
              "attributes": {
                "name": "address",
                "type": "address"
              },
```

```
        "id": 3204,
        "name": "ElementaryTypeName",
        "src": "148:7:7"
      }
    ],
    "id": 3205,
    "name": "VariableDeclaration",
    "src": "148:20:7"
  },
  {
    "attributes": {
      "constant": false,
      "indexed": true,
      "name": "spender",
      "scope": 3211,
      "stateVariable": false,
      "storageLocation": "default",
      "type": "address",
      "value": null,
      "visibility": "internal"
    },
    "children": [
      {
        "attributes": {
          "name": "address",
          "type": "address"
        },
        "id": 3206,
        "name": "ElementaryTypeName",
        "src": "170:7:7"
      }
    ],
    "id": 3207,
    "name": "VariableDeclaration",
    "src": "170:23:7"
  },
  {
    "attributes": {
      "constant": false,
      "indexed": false,
      "name": "value",
      "scope": 3211,
      "stateVariable": false,
      "storageLocation": "default",
      "type": "uint256",
      "value": null,
      "visibility": "internal"
```

```
            },
            "children": [
              {
                "attributes": {
                  "name": "uint256",
                  "type": "uint256"
                },
                "id": 3208,
                "name": "ElementaryTypeName",
                "src": "195:7:7"
              }
            ],
            "id": 3209,
            "name": "VariableDeclaration",
            "src": "195:13:7"
          }
        ],
        "id": 3210,
        "name": "ParameterList",
        "src": "147:62:7"
      }
    ],
    "id": 3211,
    "name": "EventDefinition",
    "src": "133:77:7"
  },
  {
    "attributes": {
      "constant": false,
      "name": "symbol",
      "scope": 3269,
      "stateVariable": true,
      "storageLocation": "default",
      "type": "string storage ref",
      "value": null,
      "visibility": "public"
    },
    "children": [
      {
        "attributes": {
          "name": "string",
          "type": "string storage pointer"
        },
        "id": 3212,
        "name": "ElementaryTypeName",
        "src": "215:6:7"
      }
```

```
  ],
  "id": 3213,
  "name": "VariableDeclaration",
  "src": "215:20:7"
},
{
  "attributes": {
    "body": null,
    "constant": true,
    "implemented": false,
    "isConstructor": false,
    "modifiers": [
      null
    ],
    "name": "decimals",
    "payable": false,
    "scope": 3269,
    "stateMutability": "view",
    "superFunction": null,
    "visibility": "public"
  },
  "children": [
    {
      "attributes": {
        "parameters": [
          null
        ]
      },
      "children": [],
      "id": 3214,
      "name": "ParameterList",
      "src": "259:2:7"
    },
    {
      "children": [
        {
          "attributes": {
            "constant": false,
            "name": "",
            "scope": 3218,
            "stateVariable": false,
            "storageLocation": "default",
            "type": "uint8",
            "value": null,
            "visibility": "internal"
          },
          "children": [
```

```
            {
              "attributes": {
                "name": "uint8",
                "type": "uint8"
              },
              "id": 3215,
              "name": "ElementaryTypeName",
              "src": "280:5:7"
            }
          ],
          "id": 3216,
          "name": "VariableDeclaration",
          "src": "280:5:7"
        }
      ],
      "id": 3217,
      "name": "ParameterList",
      "src": "279:7:7"
    }
  ],
  "id": 3218,
  "name": "FunctionDefinition",
  "src": "242:45:7"
},
{
  "attributes": {
    "body": null,
    "constant": true,
    "implemented": false,
    "isConstructor": false,
    "modifiers": [
      null
    ],
    "name": "totalSupply",
    "payable": false,
    "scope": 3269,
    "stateMutability": "view",
    "superFunction": null,
    "visibility": "public"
  },
  "children": [
    {
      "attributes": {
        "parameters": [
          null
        ]
      },
```

```
        "children": [],
        "id": 3219,
        "name": "ParameterList",
        "src": "312:2:7"
      },
      {
        "children": [
          {
            "attributes": {
              "constant": false,
              "name": "supply",
              "scope": 3223,
              "stateVariable": false,
              "storageLocation": "default",
              "type": "uint256",
              "value": null,
              "visibility": "internal"
            },
            "children": [
              {
                "attributes": {
                  "name": "uint256",
                  "type": "uint256"
                },
                "id": 3220,
                "name": "ElementaryTypeName",
                "src": "333:7:7"
              }
            ],
            "id": 3221,
            "name": "VariableDeclaration",
            "src": "333:14:7"
          }
        ],
        "id": 3222,
        "name": "ParameterList",
        "src": "332:16:7"
      }
    ],
    "id": 3223,
    "name": "FunctionDefinition",
    "src": "292:57:7"
  },
  {
    "attributes": {
      "body": null,
      "constant": true,
```

```
  "implemented": false,
  "isConstructor": false,
  "modifiers": [
    null
  ],
  "name": "balanceOf",
  "payable": false,
  "scope": 3269,
  "stateMutability": "view",
  "superFunction": null,
  "visibility": "public"
},
"children": [
  {
    "children": [
      {
        "attributes": {
          "constant": false,
          "name": "_owner",
          "scope": 3230,
          "stateVariable": false,
          "storageLocation": "default",
          "type": "address",
          "value": null,
          "visibility": "internal"
        },
        "children": [
          {
            "attributes": {
              "name": "address",
              "type": "address"
            },
            "id": 3224,
            "name": "ElementaryTypeName",
            "src": "373:7:7"
          }
        ],
        "id": 3225,
        "name": "VariableDeclaration",
        "src": "373:14:7"
      }
    ],
    "id": 3226,
    "name": "ParameterList",
    "src": "372:16:7"
  },
  {
```

```
      "children": [
        {
          "attributes": {
            "constant": false,
            "name": "balance",
            "scope": 3230,
            "stateVariable": false,
            "storageLocation": "default",
            "type": "uint256",
            "value": null,
            "visibility": "internal"
          },
          "children": [
            {
              "attributes": {
                "name": "uint256",
                "type": "uint256"
              },
              "id": 3227,
              "name": "ElementaryTypeName",
              "src": "407:7:7"
            }
          ],
          "id": 3228,
          "name": "VariableDeclaration",
          "src": "407:15:7"
        }
      ],
      "id": 3229,
      "name": "ParameterList",
      "src": "406:17:7"
    }
  ],
  "id": 3230,
  "name": "FunctionDefinition",
  "src": "354:70:7"
},
{
  "attributes": {
    "body": null,
    "constant": false,
    "implemented": false,
    "isConstructor": false,
    "modifiers": [
      null
    ],
    "name": "transfer",
```

```
  "payable": false,
  "scope": 3269,
  "stateMutability": "nonpayable",
  "superFunction": null,
  "visibility": "public"
},
"children": [
  {
    "children": [
      {
        "attributes": {
          "constant": false,
          "name": "_to",
          "scope": 3239,
          "stateVariable": false,
          "storageLocation": "default",
          "type": "address",
          "value": null,
          "visibility": "internal"
        },
        "children": [
          {
            "attributes": {
              "name": "address",
              "type": "address"
            },
            "id": 3231,
            "name": "ElementaryTypeName",
            "src": "447:7:7"
          }
        ],
        "id": 3232,
        "name": "VariableDeclaration",
        "src": "447:11:7"
      },
      {
        "attributes": {
          "constant": false,
          "name": "_value",
          "scope": 3239,
          "stateVariable": false,
          "storageLocation": "default",
          "type": "uint256",
          "value": null,
          "visibility": "internal"
        },
        "children": [
```

```
          {
            "attributes": {
              "name": "uint256",
              "type": "uint256"
            },
            "id": 3233,
            "name": "ElementaryTypeName",
            "src": "460:7:7"
          }
        ],
        "id": 3234,
        "name": "VariableDeclaration",
        "src": "460:14:7"
      }
    ],
    "id": 3235,
    "name": "ParameterList",
    "src": "446:29:7"
  },
  {
    "children": [
      {
        "attributes": {
          "constant": false,
          "name": "success",
          "scope": 3239,
          "stateVariable": false,
          "storageLocation": "default",
          "type": "bool",
          "value": null,
          "visibility": "internal"
        },
        "children": [
          {
            "attributes": {
              "name": "bool",
              "type": "bool"
            },
            "id": 3236,
            "name": "ElementaryTypeName",
            "src": "485:4:7"
          }
        ],
        "id": 3237,
        "name": "VariableDeclaration",
        "src": "485:12:7"
      }
```

```
      ],
      "id": 3238,
      "name": "ParameterList",
      "src": "484:14:7"
    }
  ],
  "id": 3239,
  "name": "FunctionDefinition",
  "src": "429:70:7"
},
{
  "attributes": {
    "body": null,
    "constant": false,
    "implemented": false,
    "isConstructor": false,
    "modifiers": [
      null
    ],
    "name": "transferFrom",
    "payable": false,
    "scope": 3269,
    "stateMutability": "nonpayable",
    "superFunction": null,
    "visibility": "public"
  },
  "children": [
    {
      "children": [
        {
          "attributes": {
            "constant": false,
            "name": "_from",
            "scope": 3250,
            "stateVariable": false,
            "storageLocation": "default",
            "type": "address",
            "value": null,
            "visibility": "internal"
          },
          "children": [
            {
              "attributes": {
                "name": "address",
                "type": "address"
              },
              "id": 3240,
```

```
        "name": "ElementaryTypeName",
        "src": "526:7:7"
      }
    ],
    "id": 3241,
    "name": "VariableDeclaration",
    "src": "526:13:7"
  },
  {
    "attributes": {
      "constant": false,
      "name": "_to",
      "scope": 3250,
      "stateVariable": false,
      "storageLocation": "default",
      "type": "address",
      "value": null,
      "visibility": "internal"
    },
    "children": [
      {
        "attributes": {
          "name": "address",
          "type": "address"
        },
        "id": 3242,
        "name": "ElementaryTypeName",
        "src": "541:7:7"
      }
    ],
    "id": 3243,
    "name": "VariableDeclaration",
    "src": "541:11:7"
  },
  {
    "attributes": {
      "constant": false,
      "name": "_value",
      "scope": 3250,
      "stateVariable": false,
      "storageLocation": "default",
      "type": "uint256",
      "value": null,
      "visibility": "internal"
    },
    "children": [
      {
```

```
            "attributes": {
              "name": "uint256",
              "type": "uint256"
            },
            "id": 3244,
            "name": "ElementaryTypeName",
            "src": "554:7:7"
          }
        ],
        "id": 3245,
        "name": "VariableDeclaration",
        "src": "554:14:7"
      }
    ],
    "id": 3246,
    "name": "ParameterList",
    "src": "525:44:7"
  },
  {
    "children": [
      {
        "attributes": {
          "constant": false,
          "name": "success",
          "scope": 3250,
          "stateVariable": false,
          "storageLocation": "default",
          "type": "bool",
          "value": null,
          "visibility": "internal"
        },
        "children": [
          {
            "attributes": {
              "name": "bool",
              "type": "bool"
            },
            "id": 3247,
            "name": "ElementaryTypeName",
            "src": "579:4:7"
          }
        ],
        "id": 3248,
        "name": "VariableDeclaration",
        "src": "579:12:7"
      }
    ],
```

```
      "id": 3249,
      "name": "ParameterList",
      "src": "578:14:7"
    }
  ],
  "id": 3250,
  "name": "FunctionDefinition",
  "src": "504:89:7"
},
{
  "attributes": {
    "body": null,
    "constant": false,
    "implemented": false,
    "isConstructor": false,
    "modifiers": [
      null
    ],
    "name": "approve",
    "payable": false,
    "scope": 3269,
    "stateMutability": "nonpayable",
    "superFunction": null,
    "visibility": "public"
  },
  "children": [
    {
      "children": [
        {
          "attributes": {
            "constant": false,
            "name": "_spender",
            "scope": 3259,
            "stateVariable": false,
            "storageLocation": "default",
            "type": "address",
            "value": null,
            "visibility": "internal"
          },
          "children": [
            {
              "attributes": {
                "name": "address",
                "type": "address"
              },
              "id": 3251,
              "name": "ElementaryTypeName",
```

```
            "src": "615:7:7"
          }
        ],
        "id": 3252,
        "name": "VariableDeclaration",
        "src": "615:16:7"
      },
      {
        "attributes": {
          "constant": false,
          "name": "_value",
          "scope": 3259,
          "stateVariable": false,
          "storageLocation": "default",
          "type": "uint256",
          "value": null,
          "visibility": "internal"
        },
        "children": [
          {
            "attributes": {
              "name": "uint256",
              "type": "uint256"
            },
            "id": 3253,
            "name": "ElementaryTypeName",
            "src": "633:7:7"
          }
        ],
        "id": 3254,
        "name": "VariableDeclaration",
        "src": "633:14:7"
      }
    ],
    "id": 3255,
    "name": "ParameterList",
    "src": "614:34:7"
  },
  {
    "children": [
      {
        "attributes": {
          "constant": false,
          "name": "success",
          "scope": 3259,
          "stateVariable": false,
          "storageLocation": "default",
```

```
            "type": "bool",
            "value": null,
            "visibility": "internal"
          },
          "children": [
            {
              "attributes": {
                "name": "bool",
                "type": "bool"
              },
              "id": 3256,
              "name": "ElementaryTypeName",
              "src": "658:4:7"
            }
          ],
          "id": 3257,
          "name": "VariableDeclaration",
          "src": "658:12:7"
        }
      ],
      "id": 3258,
      "name": "ParameterList",
      "src": "657:14:7"
    }
  ],
  "id": 3259,
  "name": "FunctionDefinition",
  "src": "598:74:7"
},
{
  "attributes": {
    "body": null,
    "constant": true,
    "implemented": false,
    "isConstructor": false,
    "modifiers": [
      null
    ],
    "name": "allowance",
    "payable": false,
    "scope": 3269,
    "stateMutability": "view",
    "superFunction": null,
    "visibility": "public"
  },
  "children": [
    {
```

```
"children": [
  {
    "attributes": {
      "constant": false,
      "name": "_owner",
      "scope": 3268,
      "stateVariable": false,
      "storageLocation": "default",
      "type": "address",
      "value": null,
      "visibility": "internal"
    },
    "children": [
      {
        "attributes": {
          "name": "address",
          "type": "address"
        },
        "id": 3260,
        "name": "ElementaryTypeName",
        "src": "696:7:7"
      }
    ],
    "id": 3261,
    "name": "VariableDeclaration",
    "src": "696:14:7"
  },
  {
    "attributes": {
      "constant": false,
      "name": "_spender",
      "scope": 3268,
      "stateVariable": false,
      "storageLocation": "default",
      "type": "address",
      "value": null,
      "visibility": "internal"
    },
    "children": [
      {
        "attributes": {
          "name": "address",
          "type": "address"
        },
        "id": 3262,
        "name": "ElementaryTypeName",
        "src": "712:7:7"
```

```
            }
          ],
          "id": 3263,
          "name": "VariableDeclaration",
          "src": "712:16:7"
        }
      ],
      "id": 3264,
      "name": "ParameterList",
      "src": "695:34:7"
    },
    {
      "children": [
        {
          "attributes": {
            "constant": false,
            "name": "remaining",
            "scope": 3268,
            "stateVariable": false,
            "storageLocation": "default",
            "type": "uint256",
            "value": null,
            "visibility": "internal"
          },
          "children": [
            {
              "attributes": {
                "name": "uint256",
                "type": "uint256"
              },
              "id": 3265,
              "name": "ElementaryTypeName",
              "src": "748:7:7"
            }
          ],
          "id": 3266,
          "name": "VariableDeclaration",
          "src": "748:17:7"
        }
      ],
      "id": 3267,
      "name": "ParameterList",
      "src": "747:19:7"
    }
  ],
  "id": 3268,
  "name": "FunctionDefinition",
```

```
                "src": "677:90:7"
              }
            ],
            "id": 3269,
            "name": "ContractDefinition",
            "src": "26:743:7"
          }
        ],
        "id": 3270,
        "name": "SourceUnit",
        "src": "0:770:7"
      },
      "compiler": {
        "name": "solc",
        "version": "0.4.19+commit.c4cbbb05.Emscripten.clang"
      },
```


ERC20Interface.json